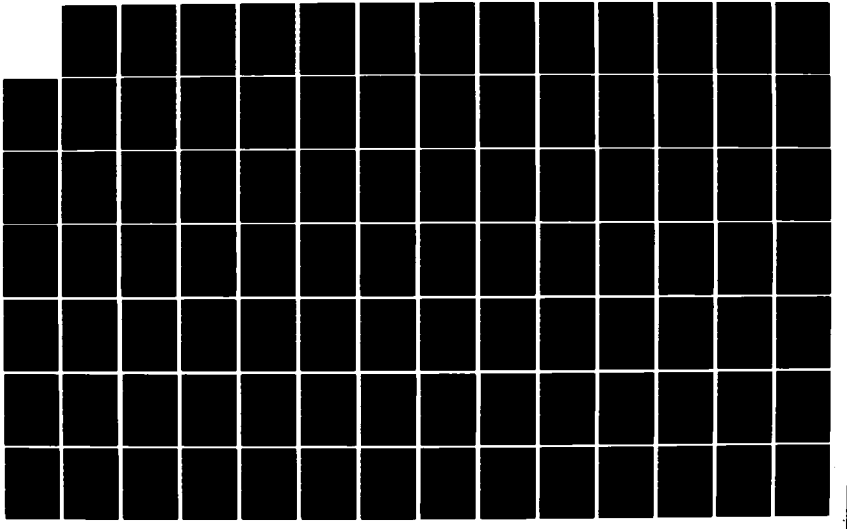
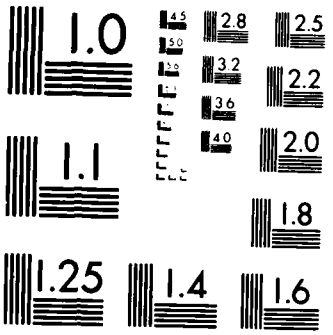
AD-A151 911 PERSONAL COMPUTER AIDED DECISION ANALYSIS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING G R WHITE 14 DEC 84 AFIT/GSO/OS/84D-8 1/3

UNCLASSIFIED F/G 9/2 NL





AFIT/GSO/OS/84D-6

AD-A151 911

PERSONAL COMPUTER AIDED DECISION ANALYSIS

THESIS

AFIT/GSO/OS/84D-6

Greg R. White

DTIC FILE COPY

Approved for public release; distribution unlimited

AFIT/GSO/OS/84D-8

PERSONAL COMPUTER AIDED DECISION ANALYSIS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Space Operations

Greg R. White, B.S.

December 1984

Preface

This study was undertaken to provide a user-friendly software tool for multiobjective decision analysis that would be of use to managers and analysts using personal computers. I sincerely hope that users will find the program helpful. Any user who would like to obtain a copy of the software or who would like to submit recommendations or suggestions for program modification should contact me at:

Greg R. White m/s 8C-21
c/o The Boeing Company
P. O. Box 3707
Seattle, Washington 98124
(206) 773-9435

I would like to thank my thesis advisor Lt Col Mark M. Mekaru for his guidance and support throughout the AFIT program and especially during this thesis research. Lt Col Mekaru and Major James K. Feldman, who participated as thesis reader, have invested great time and effort to insure that this thesis effort was a learning experience and that a valuable software tool was produced.

I also wish to thank my wife Cheryl for her typing/editing assistance, moral support and unending patience during this time. Finally, thanks go to my daughter Sara, age 4, for her loving assistance.

Greg R. White

CONTENTS

Preface ii

List of Figures iv

Abstract v

I INTRODUCTION 1

Problem Statement and Scope 6
Research Question 6
Presentation Outline 7

II LITERATURE REVIEW 8

Personal Computer Applicability to Operations Research 8
Multiobjective Decision Analysis 10

III WEIGHTING/CONSTRAINT TECHNIQUE DESCRIPTION 13

Example Multiobjective Problem 13
The Weighting Technique 16
The Constraint Technique 20

IV PROGRAM DEVELOPMENT CONSIDERATIONS 24

User Interface 24
Hardware 29
Support Software Environment 31

V IMPLEMENTATION DESCRIPTION 36

Hardware 36
Support Software 38
Program Description 39
User Interface 43
Design Innovations 43
Performance Data 48
System Outputs 49

VI CONCLUSIONS AND RECOMMENDATIONS 52

Bibliography 53

Appendix A: User's Manual 56

Appendix B: Programmer's Manual 104

Vita 265

List of Figures

Figure		
1	Problem Feasible Region	15
2	Single Objective Solution	16
3	Problem Solution/First Three Iterations	19
4	Problem Solution/Last Three Iterations	19
5	Feasible Region/Constraint Technique	21
6	System Configuration	37
7	PCADA Functional Organization	40
8	Problem Description Display	50
9	Objective Values Summary Display	50
10	Detailed Solution Display	51
11	Sample Problem Display	64
12	Objective Value Summary	100
13	Detailed Solution Display	101
14	Top Level Subroutine Tree	127
15	Define New Problem - Subroutine Tree	128
16	Problem Modification - Subroutine Tree	129
17	Problem Solution - Subroutine Tree	130
18	Subroutine Cross Reference Map	131

Abstract

The increasing complexity of today's business and military decisions demand informed decision making at all levels of management. Such decision making must be fully supported by timely and accurate analysis. Computers are well-suited for such analysis. Unfortunately, the large mainframe computers are not flexible or responsive enough for use by most managers in a timely manner.

The growing popularity, presence, and capability of microcomputers represents a new opportunity for operations research. These small, low-cost machines can provide much of the computer support needed for decision making by managers and analysts provided that the necessary software tools are developed.

This study was undertaken to provide a user-oriented decision analysis tool which exploits the advantages of personal computers. Of the many useful quantitative techniques available, the weighting and constraint techniques of multiobjective decision analysis were selected and implemented.

PERSONAL COMPUTER AIDED

DECISION ANALYSIS

I Introduction

The impact of microcomputers on management decision making has been and will continue to be complex and far reaching (30:921). As the cost of hardware declines, and the variety of available software increases, more and more managers use and rely on microcomputers on a daily basis. Unfortunately, the use of the microcomputer as a management tool has, to date, been limited mainly to financial management and to the bookkeeping needs of smaller firms (20:297). The more complex operations research techniques have, for the most part, remained inaccessible to the line manager. They exist as parts of complicated software packages on large-scale computers that usually require both operations research department and data processing department assistance to access. This inaccessibility has tended to frustrate managers and has generally inhibited the application of operations research techniques to real world problems (16:16). As a result, many feel that operations research has failed to reach it's potential, beyond an academic environment (1:93).

The first microcomputers were not much help since they

did not possess the computing power necessary for complex operations research techniques. However, over the past thirteen years, since the first microprocessor was introduced (5:198), technology has improved microcomputer capabilities such that most operations research techniques can readily be handled by a majority of today's most commonly available personal computers.

These two developments, the widespread acceptance of microcomputers and the rapid increase in microcomputer capabilities, represent a new opportunity for operations research to fulfill it's potential in a non-academic environment (11:251). However, for this to occur, software tools must be developed that provide business and military leaders the means to use their microcomputers to apply operations research techniques to real world problems. Already, software vendors, in an attempt to meet expanding needs, are beginning to incorporate some operations research technology in their new offerings. For example, project management and inventory control are already commercially available (16:17). To a degree, this commercial software development is healthy, however, the introduction of these tools as software innovations rather than MS/OR innovations leads to the risks usually associated with loss of quality control. Poorly designed tools will be resisted by management and will do little to improve the already weak academic/management relationship (16:17).

Magnifying the importance of applying operations research techniques to microcomputers is the increasing complexity of today's business and military decisions. In addition, the consequences of making incorrect decisions are frequently so severe that better decision making capabilities are becoming increasingly essential. Operations research techniques and today's personal computers provide one means of improving management's decision making ability. The limited inventory of effective, manager oriented software tools is the major obstacle which prevents realization of this goal. An analysis of the decision making process helps to reveal the next logical place where microcomputer implementation efforts should be directed.

Decision making is an essential aspect of all of our daily lives (6:4). Even the simplest situations involve the decision making process. What clothes to wear; What food to eat; or what activities to undertake, are but a small sample of the thousands of decisions we all make each day. But how do we make these decisions? In general, our decisions are made to optimize some condition based on an analysis of the situation (31:14).

For example, in deciding what clothes to wear, we may base our decision on: what is clean, what the weather is like, and/or what type of activities we plan to undertake. In general, some subset of our wardrobe will be appropriate for the situation while the rest will not. That is, some of

our clothes will be dirty, or not warm enough, or not "fancy" enough, etc. Once this acceptable subset of our wardrobe has been established, we then select that outfit which optimizes some situation (or set of conditions). Examples of the conditions we may wish to optimize in this case, would be:

"to look the best", or

"to be the most comfortable".

The situation characteristics that establish "what is appropriate" are usually referred to as the constraints (31:570). The condition we are attempting to optimize is usually referred to as an objective (31:572). Thus the decision process involves optimizing some objective subject to one or more constraints (31:43).

Of course, for most of our everyday decisions we do not consciously enumerate constraints and objectives. For most of these decisions, we rely on our instincts or intuition. However, as the problem grows in importance, complexity or size we can no longer rely on our intuition. For these decisions, operations research decision making aids such as linear programming can help us make the best decisions. Several successful microcomputer applications of linear programming have previously been demonstrated (10, 12).

The decision making task is further complicated when we wish to optimize two or more objectives simultaneously.

Usually, with these multiobjective problems improving one objective results in hurting at least one of the other objectives (6:16). The objectives usually compete for the same scarce resources.

These types of multiobjective problems are common in today's fast-paced and complex society. Both business and military leaders are faced with these types of problems daily. The consequences of suboptimal decisions for these problems can be far-reaching and in some cases catastrophic (31:4). A microcomputer aid for dealing with multiobjective decision problems would be a useful decision making tool. A literature survey has revealed no such application to date.

Microcomputer implementation of a multiobjective decision tool is a problem with two dimensions. It is indeed a software problem in that the complexity of the techniques to be programmed will require taking full advantage of the computer's capabilities in order to run at all (27:294). But more importantly, the problem has an operations research orientation. That is, the key to success of the program is the creation of a problem solving environment that can be used directly by the manager in addition to the operations research staff (16:18). Thus, rather than simply an issue of correctly programming a technique, the problem involves technique selection and design, user interface, and user friendliness. All of these latter issues can best be resolved by the operations research specialist (18:275).

Problem Statement and Scope

The increasing costs of incorrect decisions have implied a need to improve the decision making aids available to industrial and military leaders. Operations research and the availability of microcomputers have provided the means to do so. Successful implementation of operational research tools, such that managers will embrace their use, is the problem that needs to be resolved in order to achieve the goal. One such tool, Multiobjective Decision Analysis, has not yet been applied to the microcomputer environment.

This research involves implementation of the constraint and weighting multiobjective analysis techniques (8:211). The program is designed and implemented in such a way to also serve as an instructional aid to students and/or inexperienced users. Hardware, software, and user characteristics were considered in order to maximize the utility of the resultant package. Special emphasis was placed on program portability and user friendliness. Structured design techniques were employed to insure that the end product is maintainable.

Research Question

Consideration of the problem described above leads to the following questions: Can a multiobjective decision analysis package be successfully implemented on a personal computer? Would such an implementation be usable in a problem solving environment without operations research staff

and/or data processing staff support? Would such an application improve the manager's decision making ability? With these questions in mind, the overall objective of this research was to develop a multiobjective decision analysis package for a personal computer. Specific subobjectives were:

1) To maximize program portability through proper hardware selection, implementation language selection, and program design.

2) To maximize program acceptance to users regardless of their expertise through a "friendly" user interface.

3) To maximize program maintainability by utilizing a top-down design methodology and by providing complete user and programmer documentation.

Presentation Outline

Section II describes the current state-of-the-art for the primary research question, namely: Can multiobjective decision techniques be successfully applied to a microcomputer? Section III provides a background description for the techniques that were implemented. The important program development considerations are presented in section IV, with a specific description of the implementation provided in section V. Finally, research conclusions and recommendations appear in section VI. Program documentation consisting of a user's manual and programmer's manual appear as Appendix A and Appendix B respectively.

II Literature Review

In order to successfully complete this research, a literature review of the pertinent areas was performed and is summarized below.

Personal Computer Applicability to Operations Research

A number of authors have recognized the microcomputer's applicability to operations research problems (4, 11, 16, 18, 20, 30). In Hollock's 1981 article (18), the increasing importance that personal computers are expected to play in operations research is sighted, along with a description of a specific successful application in the steel industry. Douglas also emphasizes the increasing influence of personal computers in his article (11). However, he is particularly enthusiastic about the fact that personal computers provide the means to bring the problem solving process closer to the client (11:253). In addition, he points out the potential to use color and/or graphics in improving operational research's ability to describe problem solutions (11:254). In Geoffrion (16), the author describes the MS/OR field as consisting of two separate domains: The world of action and affairs, epitomized by line managers; and, the world of ideas, epitomized by academics(16:11). The author describes what he preceives to be a considerable rift between the two domains, and cites the personal computer as the factor most likely to eliminate the differences between the two (16:14).

The articles described above justify the need for microcomputer applications to operations research. Other works begin by assuming that this need exists, and then proceed to describe the wide range of hardware configurations and options available for application. Two articles and one book in particular, define what microcomputers are, and then compare/contrast the various hardware configurations (5, 7, 27). In an article by Ranyard (27), the author describes the characteristics that are desirable in personal computers which are to be applied to operations research problems (27:281). He then describes four companies' experiences with personal computers in their operations research departments. The computers compared were: Cromemco, PETS, TEI PT212, and PRIME (27:291). These computers are all similar and are typical of many of today's average microcomputers (i.e. 64K of memory, 0.5 Megabyte disc storage, 8 or 16 bit central processor, printer, and video display). The companies met with varying levels of success, however, in general, they were all satisfied with the various systems. The article by Clementson and Clewett (7), and the book by Cassell (5), describe available microcomputer features in more general, less operations research specific terms.

Two thesis (10, 12) that demonstrate linear programming applications to personal computers have been reviewed. Although both programs perform principally the same function, they differ considerably in implementation and ease of use.

In the thesis by Fraley and Kem (12), a FORTRAN implementation for an APPLE II+ computer is demonstrated. The program provides relatively rapid response and a comprehensive linear programming package, but is somewhat difficult to use because of limitations inherent in the PASCAL operating system (which is required in order to run FORTRAN on an APPLE computer). Conte, in his 1979 research, demonstrated a BASIC implementation on an APPLE II+ computer. The program provides a basic linear programming capability but is somewhat slow in response due to the speed limitations inherent in interpretive BASIC. These two different implementations of a similar package demonstrate the kinds of tradeoffs necessary when considering a microcomputer implementation.

The special design considerations for programs that are to be implemented on microcomputers are documented in a book by Freedman and Evans (13). Emphasis is given to a systematic approach that begins with the system's functional specification and proceeds through system tests. Although not directly oriented towards operations research type problems, the Freedman book defines those steps necessary to insure successful microcomputer program implementation.

Multiobjective Decision Analysis

This phase of the literature review involved investigating techniques appropriate for implementation in this

research. One fairly recent article by Cohon and Marks (8) provides a comprehensive review of existing multiobjective analysis techniques. In order to evaluate the various techniques, the authors established three different evaluation criteria:

a) computational efficiency,

b) explicitness of tradeoffs among objectives, and

c) the amount of information generated for decision making (8:208).

The article classified the various techniques into three categories:

a) generation techniques (including the weighting and constraint methods),

b) techniques which rely on prior articulation of preferences (including goal programming), and,

c) techniques which rely on progressive articulation of preferences (including the iterative weighting method) (8:210).

The authors found that the comparison of techniques was itself a multiobjective problem with three criteria. As might be expected, computational efficiency suffers as the amount of information generated increases (8:217). The authors concluded that the generating techniques were most desirable when they could be afforded in terms of computational efficiency. However, they felt that when there are four or more objectives, a technique that limits the size of

the feasible region should be used (8:219).

This research does focus on the generating techniques, specifically the weighting and constraint techniques. It leaves the other techniques for future program enhancement.

The theoretical and mathematical basis for the various techniques described above are provided in several multi-objective analysis textbooks. One excellent example is a text by Cohon (9), which presents most of the data necessary for computer implementation of the various operations research algorithms. This background data is summarized in the following section.

III Weighting/Constraint Technique Description

Techniques for solving multiobjective problems can not be characterized as new, since Kuhn and Tucker were credited with their discovery in 1951 (8:208). However, as stated earlier, the problem lies in making these techniques available, in an easy to use fashion, to the decision maker. Two techniques, the weighting and constraint techniques, have been implemeted on a personal computer as part of this research. The remainder of this section is devoted to describing these techniques. However, before continuing, a simple hypothetical multiobjective problem is presented to help clarify the description of the techniques.

Example Multiobjective Problem

Suppose some procurement office has been tasked with deciding how many each of two types of aircraft to obtain. The aircraft types are designated X1 and X2. Both aircraft are fighters offering some bombing capability. The procurement office is faced with two objectives:

a) to maximize bombing capability, and

b) to maximize fighter capability.

It has been determined that aircraft X1 is more effective as a bomber while aircraft X2 is a better fighter, and the following relationships have been established:

bomber capability = 2 times the number of X1 aircraft
+ 1 times the number of X2 aircraft (1)

fighter capability = 1 times the number of X1 aircraft
+ 2 times the number of X2 aircraft (2)

More formally, in this problem we wish to:

MAXIMIZE: 2(X1) + 1(X2) -bomber capability- (3)

MAXIMIZE: 1(X1) + 2(X2) -fighter capability- (4)

At first glance, it appears that we can obtain as high a value as desired for both bomber and fighter capabilites simply by buying large numbers of aircraft. Unfortunately this type of solution is not usually practical since resources are usually scarce. For our problem we will assume that the manufacturer of X1 can supply no more than ten aircraft and the manufacturer of X2 can supply no more than eight aircraft. A higher authority has mandated that no more than a total of 12 aircraft may be procurred. These restrictions serve as the problem constraints. More formally stated, these constraints are:

X1 <= 10 (X1 supply limit) (5)
X2 <= 8 (X2 supply limit) (6)
X1 + X2 <= 12 (total procurement limit) (7)

A "non-negativity constraint" is also added to the problem since it does not make sense to procure negative numbers of aircraft. That is:

X1 and X2 >= 0 (non-negativity constraint) (8)

When dealing with problems with only two decision variables (i.e., number of X1s and number of X2s) it is helpful to use graphs to define the area of acceptable solutions. For example, Figure 1 is a graph of the problem constraints. The shaded region represents the complete set of solutions that satisfy all of the constraints. It is referred to as the feasible region.

Unfortunately, the problem involves more than just selecting a legitimate solution, it also involves selecting the solution which optimizes the objectives. With our example problem, as is usually the case, the solution which optimizes each of the objectives is not the same. Instead, we will obtain a set of solutions from which the decision maker will make his selection. Techniques for obtaining these solutions are called generating techniques (8:210) and are described in what follows.

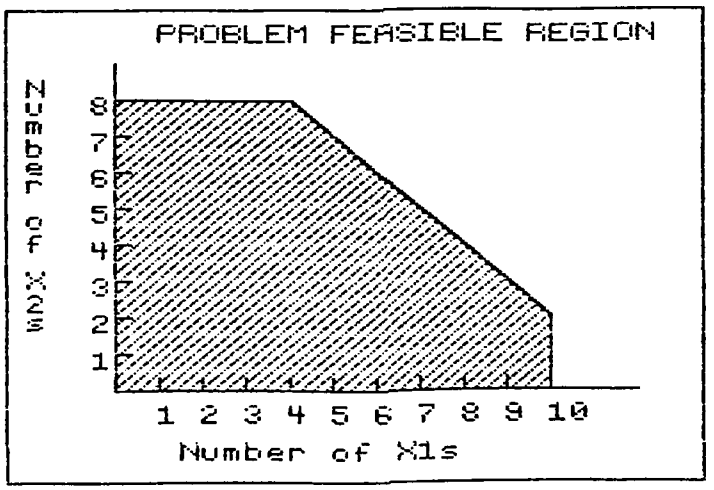


Figure 1: Problem Feasible Region

The Weighting Technique

In single objective optimization problems, the optimal solution is that feasible solution which maximizes (or minimizes) the objective. This solution can be seen graphically in a two variable problem as the point at which the objective function is just tangent to the feasible region. For example, suppose for our problem we ignore the second objective and attempt to optimize only the first. As the objective function assumes higher and higher values of "bomber capability" we see in Figure 2 that at a value of 22 (ie 2X1 + 1X2 = 22) the function is just tangent to the feasible region at point A. At higher values of bombing capability the objective function does not intersect the feasible region. That is, given the constraints, we can not achieve a higher bombing capability than 22.

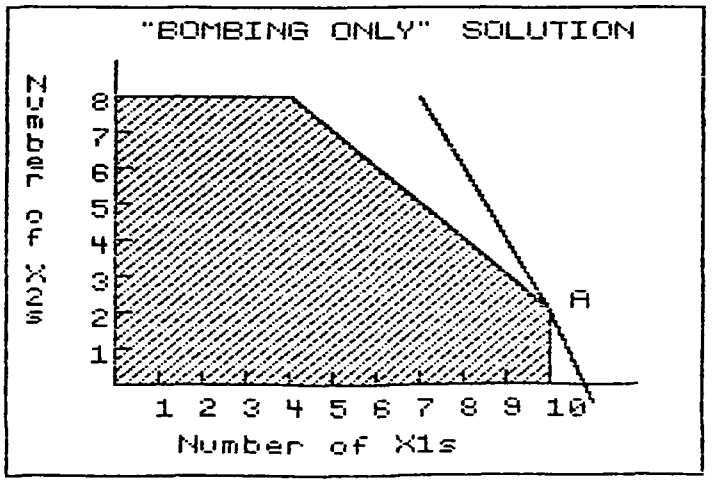


Figure 2: Single Objective Solution

The Weighting Technique

In single objective optimization problems, the optimal solution is that feasible solution which maximizes (or minimizes) the objective. This solution can be seen graphically in a two variable problem as the point at which the objective function is just tangent to the feasible region. For example, suppose for our problem we ignore the second objective and attempt to optimize only the first. As the objective function assumes higher and higher values of "bomber capability" we see in Figure 2 that at a value of 22 (ie 2X1 + 1X2 = 22) the function is just tangent to the feasible region at point A. At higher values of bombing capability the objective function does not intersect the feasible region. That is, given the constraints, we can not achieve a higher bombing capability than 22.

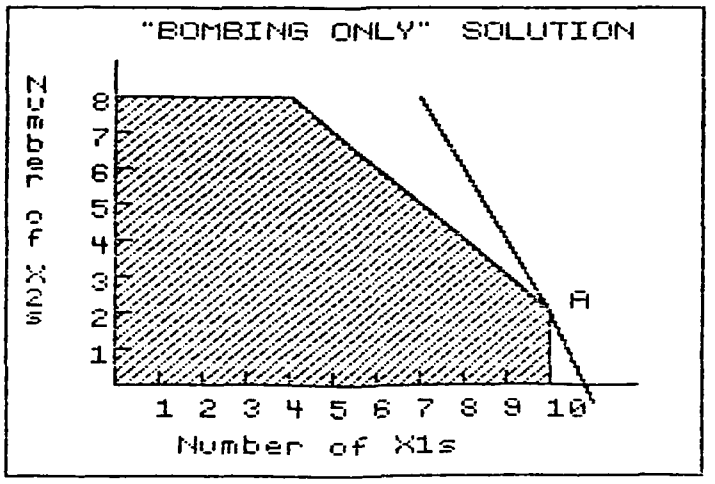


Figure 2: Single Objective Solution

The weighting technique involves solving a series of single objective problems. In each problem we use a different objective that is formed by multiplying each of the original objectives by a weight then summing them. For each solution iteration, we use a different set of weights. With this process we are solving a different objective each iteration. The solution found represents the optimal solution for that aggregate objective. Each solution then corresponds to differing levels of importance ascribed to each of the problem's objectives.

For the present example, we will solve the problem six times using objective weights ranging from one to zero as follows:

iteration	objective weights obj 1	obj 2	aggregate objective
1	1.0	0.0	2.0X1 + 1.0X2
2	0.8	0.2	1.8X1 + 1.2X2
3	0.6	0.4	1.6X1 + 1.4X2
4	0.4	0.6	1.4X1 + 1.6X2
5	0.2	0.8	1.2X1 + 1.8X2
6	0.0	1.0	1.0X1 + 2.0X2

As we move from iteration number one to iteration number six, the slope of the aggregate objective function slowly changes. This reflects the fact that as the weights that were used to make up the aggregate objective change, the emphasis for that iteration solution will also change. If we find the optimal point for each of these aggregate objective functions as we did before, by locating the point

on the highest valued objective function that is just tangent to the feasible region, we see that the first three iterations result in optimal solution "A", while the last three iterations are optimal at point "B". This result is shown graphically in Figures 3 and 4 respectively.

Following is a summary of the objective values corresponding to the above solutions:

solution	objective values obj 1	obj 2
"A"	22	14
"B"	16	20

Thus we can see that as expected, we can improve the value of objective two, but only at the expense of objective one as we move from solution A to solution B. For each of these solutions, no point offers a better value for both of the objective functions. These solutions are referred to as "non-dominated solutions."

In general, the weighting technique will generate only solutions which are corner points of the problem's feasible region. However, all of the points between the extremes (points A and B in this case) are part of the non-dominated solution set. For further information or proof of this result, the reader is referred to the text on the subject by Chankong and Haimes (6).

To summarize, the weighting technique reduces any

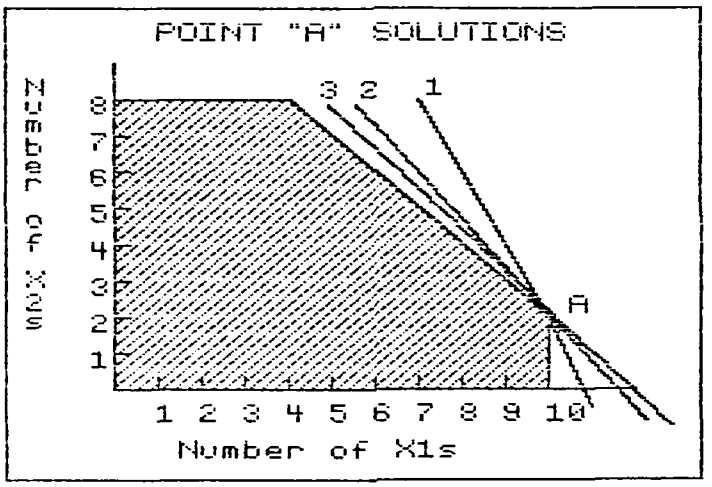


Figure 3: Problem Solution/First Three Iterations

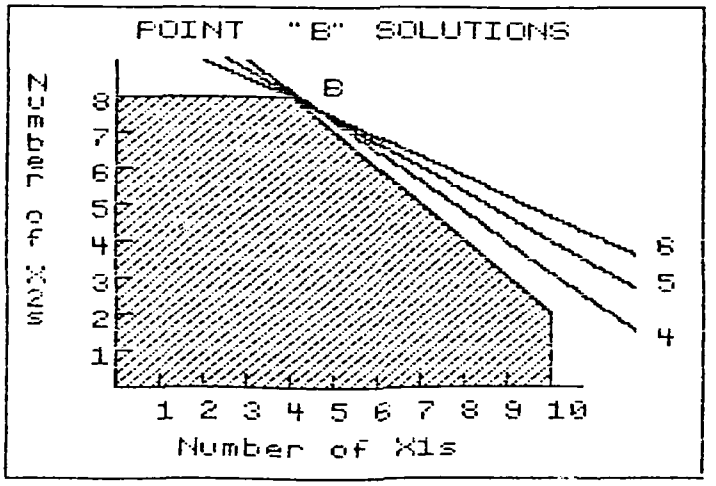


Figure 4: Problem Solution/Last Three Iterations

multiobjective problem to a series of single objective problems. Each of these generates its own optimal solution. Each optimal solution corresponds to assigning certain importance to each of the objectives in the problem. From the complete set of (non-dominated) solutions, the decision maker will select that solution which corresponds to the priorities that he assigns to each of the respective objectives. If he is unsure of how he values each objective, he can select from the non-dominated solutions, that solution which provides the best acceptable level for each objective.

The Constraint Technique

The constraint technique is similar to the weighting technique in that it reduces the multiobjective problem to a series of single objective problems. It is unlike the weighting technique in that each single objective problem uses the same objective with different constraints. Specifically, all but the one of the objective functions are added to the problem as equality constraints, each with an assigned "acceptable" objective limit. It is this objective limit that is varied for each iteration. The net result of adding objectives to the problem as constraints is to further restrict the feasible region.

For example, in the previous section we determined that for the set of non-dominated solutions, the value of the second objective ranged from 14 to 20. Suppose we determine that a fighter capability value of 17 is acceptable. With

the constraint technique then, the problem would be to maximize the bomber capability given that the fighter capability equals 17. That is, we have added the following constraint:

$$X1 + 2X2 = 17 \qquad \text{(objective \#2 limit)} \qquad (9)$$

Since this is an equality constraint, the feasible region is reduced to those points on the line defined by equation (9) that are within the original feasible region. Graphically, the new feasible region is shown in Figure 5 as all points on the line between points "D" and "C".

Given this new feasible region, the optimal value of the first objective occurs when it intersects point "C". Thus we have obtained one non-dominated solution via the constraint technique. More solutions are obtained by

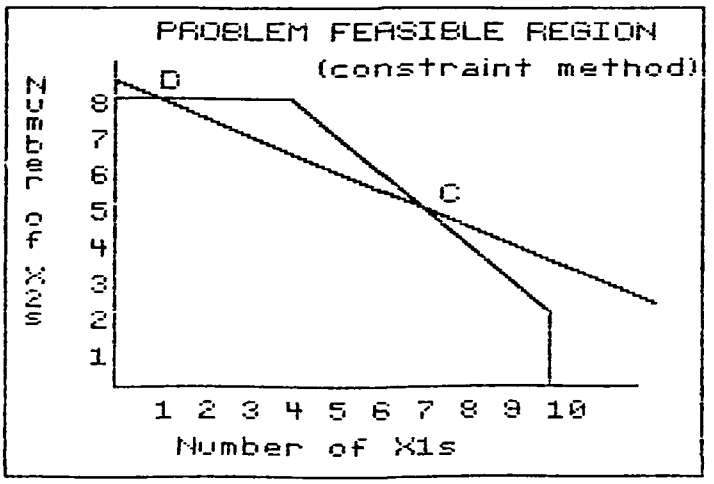


Figure 5: Feasible Region/Constraint Technique

iteratively solving the problem varying only the limit on the objectives that now serve as constraints.

In this case, if we vary the objective limit from 14 to 20 in seven steps we will obtain seven different non-dominated solutions as follows:

objective #2 limiting value	objective #1 computed value	decision variables X1	X2
14	22	10	2
15	21	9	3
16	20	8	4
17	19	7	5
18	18	6	6
19	17	5	7
20	16	6	8

As before, we see that we can achieve an improvement in one objective, but only at the expense of the other objective.

The constraint technique has the advantage that it will generate more than just corner point solutions of the original feasible region. This technique does however require some prior knowledge of the range that the objectives can realistically be expected to assume. That is, if we were to set the level of objective two at any value over 20, we would obtain no feasible solutions. Similarly, setting objective two to any value under 14 would result in a solution with a suboptimal value for objective one. In a sense, all iterations that set the objective two limit outside of the 14 to 20 range are wasted since they will not generate a non-dominated solution. Further information concerning this technique can be found in the text by

Chankong and Haimes (6).

To summarize, the constraint technique involves reducing a multiobjective problem to a series of single objective problems. Each single objective problem uses one of the original objectives as its objective and adds the others to the problem as equality constraints. Each iteration involves setting the "acceptable" level to different values for each of the objective function equality constraints. After all iterations, the decision maker will select from the set of non-dominated solutions in the same fashion as for the weighting technique.

IV Program Development Considerations

Development of a Multiobjective Decision Analysis package required a careful analysis of the end user, the hardware, and the support software environment prior to the program design process. This section describes program considerations related to these three topics.

User Interface

How the program is accepted by the end user is considered to be the most important criterion for determining program success. Two design aspects needed to be addressed in order to insure user acceptance. First, it was felt that the system must increase the timeliness, effectiveness, and availability of information (28:66); and second, a friendly interface that does not require its users to be a specialist in either operations research or computer science was deemed essential.

Systems that satisfy the first goal are referred to as "quick response systems." These systems offer the following advantages:

-- they allow managers to react more rapidly

-- they reduce waste in the use of resources

-- they permit quick follow-up on creative ideas (28:67).

It was determined that the quick response goal would be achieved automatically as a result of hosting the package on

IV Program Development Considerations

Development of a Multiobjective Decision Analysis package required a careful analysis of the end user, the hardware, and the support software environment prior to the program design process. This section describes program considerations related to these three topics.

User Interface

How the program is accepted by the end user is considered to be the most important criterion for determining program success. Two design aspects needed to be addressed in order to insure user acceptance. First, it was felt that the system must increase the timeliness, effectiveness, and availability of information (28:66); and second, a friendly interface that does not require its users to be a specialist in either operations research or computer science was deemed essential.

Systems that satisfy the first goal are referred to as "quick response systems." These systems offer the following advantages:

-- they allow managers to react more rapidly

-- they reduce waste in the use of resources

-- they permit quick follow-up on creative ideas (28:67).

It was determined that the quick response goal would be achieved automatically as a result of hosting the package on

a microcomputer. The result would be a software package that is readily available on the managers desk. Its use would not be subject to the data processing department's schedules.

The second goal of a user friendly system though was not as easy to obtain. User friendliness in computer systems involves much more than easy to read system prompts (21:125). It requires a careful analysis of the end users needs and characteristics (29:125). Throughout this research, seven practical steps were employed to help insure the development of a truly user friendly system.

Minimum worker effort. It was determined that the system should require minimum worker effort of its users (14:165). Workers should be required to perform only work that is absolutely essential and can not be performed by the system. For example, in this system, although the software is only capable of maximizing objectives, it allows users to enter objectives to be minimized. Rather then requiring it of the user, the program converts these objectives (by multiplying by -1) to a form to be maximized. All displays indicate the data in the form as entered. The fact that the data had been internally transformed is of no consequence to the user and is kept transparent to him.

Another technique employed to minimize worker effort is to not require workers to repeat work done in the past. This is accomplished by the program by its "remembering"

problem descriptions and by offering the means to obtain problem results in hardcopy form. In addition, the user is not required to search for system information as documentation concerning every system prompt is available on-line as part of the system's HELP capability.

Minimal worker memory. System acceptance is enhanced when system use does not depend on excessive use of the workers memory (14:165). For this system, users are not required to memorize complex command strings or non-task related terminology. Rather than a command directed program, a menu of options is given to the user for selecting the next system operation. In addition, menus are organized in a hierarchical fashion so that only the applicable options are presented at each point. Furthermore, all system prompts and error messages are complete sentences and do not utilize computer or other unfamiliar terms. Terminology is consistent throughout all software functions.

Minimum worker frustration. Systems should spare the worker frustrations that may arise from a delay in the accomplishment of a task (14:166). In this system, since it is run on a dedicated personal computer, there is no delay from step to step. If the worker is interrupted, the system provides the means for the user to review his work prior to the interruption. In most cases, recovery from erroneous menu responses is accomplished with one keystroke. Finally, menus are hierarchically organized so that users do not have

to perform or explicitly bypass unnecessary steps.

Another common source of worker frustration arises from systems that present cryptic error messages or systems that fail from actions that do not on the surface, appear to be in error. For example, on some systems, input data is not subjected to range checks. In these cases, users may input data that can later cause an overflow (such as when dividing by zero). User friendly systems should screen input data to prevent the occurrence of nonsensical conditions.

Maximizing Use of Habit Patterns. Systems should respond to the human tendency to form long and short-term patterns of action (14:166). Systems should take advantage of "muscle memory" by consistently using the same keys for specific functions, by consistently placing similar information on display screens in the same position, and by consistent design of screen and hardcopy formats. The worker should be able to accomplish tasks using a single consistent approach and terminology for all program functions.

Prompt Problem Notification. The worker should be notified of a problem as soon as it is detected and be notified of potential problems, wherever possible, in advance of their occurrence (14:166). As mentioned earlier, careful screening of inputs at the time of entry rather than at the time of use can prevent the compounding of error conditions. Following user input errors, the user is given the opportunity to correct the problem, and control is returned to the

point of interruption. User inputs are automatically saved to disk to prevent inadvertent loss due to inexperience and/or external circumstances.

Maximum worker control of tasks. The worker should control the flow and sequence of work to the extent possible where there are no sequence-dependent activities (14:166). The worker should be able to modify the priorities of processing by the computer. Finally, the user should be able to store and retrieve information in a consistent manner. In this implementation of multiobjective decision analysis, users are free to define and/or solve problems in any sequence with the individual problem files freely interchangable from disk. The presentation order and extent of most system outputs are totally user specified as is the order of utilization of the solution techniques.

Input/Output Design Considerations. Several items related to the input/output of system information should be considered to maximize system user friendliness. First, all system outputs including questions, menus, error messages, displays and help information should be designed in an attractive way. This involves use of the entire screen width, usage of both upper and lower case characters, and the wise usage of spacing. All of these factors contribute to the ease with which outputs are interpreted. Also, care should be taken to prevent data from "scrolling off" the screen before the user has had an opportunity to interpret the

information. That is, most outputs should be designed with a screen size (24 lines by 80 characters) in mind. For displays requiring more space, the user should be in control of when the display should continue after each "page". In addition to being neat and orderly, both screen and hardcopy displays should be concise and contain only information pertinent to the present task.

Hardware

The multitude of available microprocessors (over 250 as of late 1983) and the frequency with which new ones are introduced makes it virtually impossible to be up to date on all of them (5:197). Yet, hardware selection is a critical element of any successful microcomputer application.

Since one of the primary goals of this research is to make multiobjective techniques universally available, it was decided that the implementation should necessarily involve a popular and common hardware configuration. Software which is developed on a system with limited availability to the target users will not be used extensively (12:47). Selecting a readily available system does not guarantee that the software package will be easily transportable. It is also essential to consider system modifications and required peripherals of the target system (12:47). If these modifications or added equipment are unique to the development system, widespread use of the software will be limited. Thus it was decided that the target system should also be a

basic system, that is, require only the standard assortment of peripherals.

Even with these two restrictions (i.e. commonly available system and no unique peripherals required) there is a wide range of potential target systems. An examination of the general characteristics of potential host microprocessors was necessary prior to target system selection.

Size. Microprocessors are commercially available in the following sizes: 1, 2, 4, 8, 12, 16, and 32 bits. The term size refers to the number of bits in the microprocessor's internal registers (5:212). Operationally, size determines the extent of the machine's instruction set and its addressing capability. Eight and sixteen bit microprocessors are the most common and both possess the addressing range required for the problem at hand (i.e. 65,536 or more words).

Speed. The speed of a microprocessor is determined by how fast it executes programmed instructions. Speed is greatly affected by the type of instruction (i.e. division, shift, multiplication, store, increment, etc.). Thus for most microprocessors speed is specified as a range of speeds for the complete instruction set. For eight-bit microprocessors, instruction speeds range from 0.7 to 37.5 microseconds (5:224). Because of the complexity of the multi-objective decision problem, and because systems with a slow response time are not considered user friendly, it was

determined that user acceptance would be enhanced by utilizing a relatively fast microprocessor.

Other Characteristics. There are several other characteristics of microprocessors including semi-conductor technology, ways in which the microprocessor is interfaced to memory and input/output subsystems, internal organization, addressing modes, instruction set, and the degree to which the microprocessor supports other devices (5:211). For the purpose of this research, these characteristics were not explicitly considered since the target system will be a complete system in a standard configuration. The internal characteristics of the microprocessor will be transparent to the end user.

Summarizing, we have the following requirements for the microprocessor on which the multiobjective package was to be implemented:

-- commonly available system
-- requires no unique periphreal equipment
-- relatively fast
-- capable of addressing at least 64K words
-- capable of supporting a high level language

Support Software Environment

The support software environment refers to all computer software required for program development. It includes, compilers, disk operating systems, data base management

systems, and etc. For this application, the disk operating system is taken as a given once the hardware configuration is selected since this type of software is highly machine dependent, and not portable from microprocessor to microprocessor. Also, all data manipulation/organization will be handled strictly within the multiobjective decision analysis package itself such that no external data base management system will be required. Therefore, the only decision to be made concerning the support software environment involves selection of the programming language for implementation. Once the programming language is selected, the language support tools including linkage editors and loaders will be given since these packages are generally interdependent.

Machine language programming is the most efficient in terms of program size and execution speed (15:14). However, because of the desire for program portability, a machine or assembly language implementation was not considered, since such implementations are not at all portable to microprocessors other then the original host system.

Of the high-level languages, BASIC was once the only option for microprocessors (12:48). Today however, many languages including FORTRAN, COBOL, Pascal, APL, and ADA are in common use. Because of the desire to maximize program usage through portability, it was decided that only the most widely used programming languages would be considered. This limited the selection to BASIC, FORTRAN, and Pascal (12:48).

BASIC. Most personal computers come "equipped" with the BASIC programming language. The BASIC language was first developed as any easy to learn and use programming language. It was not designed for running very large or very complex programs where program speed is important. BASIC is called an interpretive language since the computer interprets each instruction it encounters every time the program is executed. Although interpretive BASIC programs require less memory than most comparable compiled programs, they pay a considerable penalty in terms of execution time. This time penalty can be overcome by utilizing compiled BASIC, several implementations of which are now available for different microprocessors.

Advantages. The primary advantage to be gained by using BASIC for program implementation is the languages widespread use. Virtually all popular microprocessors support BASIC.

Disadvantages. Several programming functions that are necessary for the multiobjective decision analysis package are not implemented consistently between the various BASIC language versions. Specifically, disk file manipulations and report (output) formatting functions can vary greatly from machine to machine. This implies that a significant recoding effort would be required in order to rehost the package to some different computer.

Pascal. Pascal was developed in the early 1970s and

has proven to be a powerful and popular high level language (12:49). It is a compiled language and thus does not suffer the time penalties of interpretive BASIC.

Advantages. Pascal provides an extensive overlay capability that allows for the implementation and execution of large programs (12:50). In addition, the widespread usage and implementation of Pascal allows for increased potential for program portability.

Disadvantages. The Pascal programming language does not possess extensive output formatting capabilities (12:50). This shortcoming complicates the programming task when, as is the case with the multiobjective decision analysis package, carefully formatted outputs are desired.

FORTRAN. FORTRAN (FORmula TRANslator) is the most widely used high level language within the scientific programming community. It was developed, and first introduced in the early 1950s by IBM (12:50). Like Pascal, FORTRAN does not suffer the time penalties inherent in interpretive BASIC. Because of FORTRANs widespread use, there have been two attempts to standardize the language (12:50). The latest standardization effort took place from 1970 to 1977 and resulted in the language version that is available today on most microcomputers.

Advantages. The widespread usage of FORTRAN enhances maintainability and encourages portability. In addition, FORTRAN provides extensive report formatting

capabilities (12:50).

Disadvantages. Except for the newest language extensions which are not yet common to all microprocessor versions of FORTRAN, the language lacks the programming constructs that facilitate a structured programming approach. As a result, program maintainability suffers.

V Implementation Description

The user, hardware, and support software considerations described in the previous chapter have been analyzed for the purpose of establishing the target system configuration for this research. The primary selection criteria was to establish the system which offers the highest probability of achieving the research objectives as stated in chapter I. The purpose of this chapter is to describe the program implementation.

Hardware.

The hardware selection process was limited to 8-bit microprocessors for several reason. First, 8-bit microprocessors are currently more common in personal computers then the 16 or 32-bit size. Second, it was decided that should a rehost to a different size microprocessor become desirable at a later date, the process of switching to a more powerful unit would be easier than having to downscale the program for hosting on a smaller unit. Finally, it was determined that an 8-bit microprocessor would be sufficient for the task.

Of the available 8-bit microprocessors, the Z80 was selected because it is one of the most commonly used 8-bit microprocessors and because it is one of the fastest. Many of today's most popular personal computers utilize the Z80 or are directly compatable with it.

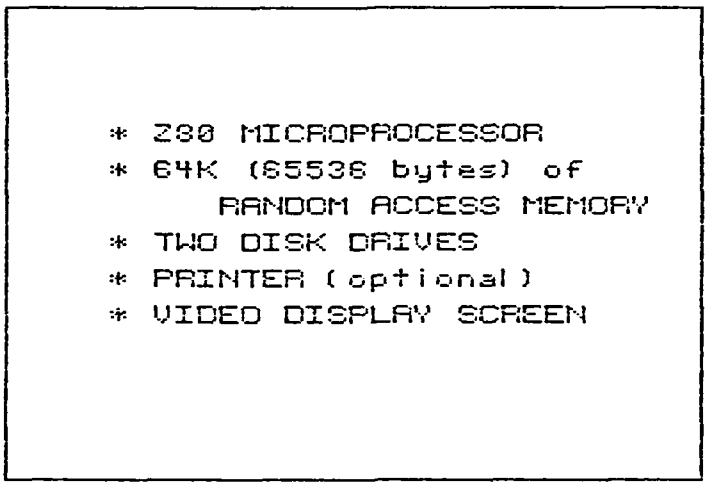


Figure 6: System Configuration

The program itself was developed on an Apple II+ computer equipped with the Z80 "softcard" by Microsoft. The program is designed to run utilizing a "standard" personal computer configuration as shown in Figure 6. The program does utilize a full 80-character screen width thus, Apple users need a special video interface (such as the Videx Videoterm) providing that capability together with a Z80 softcard.

The standard configuration (Figure 6), includes a printer which is listed as optional. This is true in the sense that the program will run without one, however, such usage is discouraged since problem results are usually lengthy and difficult to assimilate from just the video display screen.

It should also be noted that although the program was developed on an Apple computer, the implementation does not utilize any Apple unique features. In fact, the object code will run on any Z80 based computer without modification.

Support Software

As mentioned in chapter IV much of the system's support software environment is given once the hardware configuration have been selected. The selection of the Z80 microprocessor as implemented via the Apple II+ Softcard dictates the usage of the CP/M operating system for program development and operational disk input/output requirements.

The Z80 microprocessor and the CP/M operating system support all of the languages discussed in chapter IV. Nonetheless, FORTRAN was singled out as the appropriate choice mainly because of its universal portability and extensive report formatting capabilities.

It should be stressed, that as with the hardware, this program utilizes no FORTRAN or other software capabilities that are unique or specific to the Apple II+ computer. In fact, the version of FORTRAN that was used (Microsoft's FORTRAN-80) is based on the American National Standard (ANSI) FORTRAN language as described in ANSI document X3.9-1966 which was approved on March 7, 1966 (22:7). (FORTRAN-80 includes a few extensions to the language, all of which are part of the second standardization effort which was concluded in 1977.)

In short, the multiobjective decision analysis package source code can be compiled and rehosted on any personal computer that can support FORTRAN and that satisfies the configuration requirements as specified in Figure 6.

Program Description

The multiobjective decision analysis package (hereafter referred to as Personal Computer Aided Decision Analysis - PCADA) was implemented using the hardware and support software environment as described above. The program's purpose is to allow users to solve multiobjective decision problems using either the weighting or constraint techniques. The PCADA package is subject to the following limitations:

-- 4 or less objectives

-- 10 or less constraints

-- 10 or less decision variables.

Although designed for multiobjective problems, PCADA will also handle and solve single objective problems, using the same software functions.

The PCADA package consists of a main program and 23 subroutines. A total of 3,424 lines of FORTRAN code was generated to accomplish the task. The program was written in a structured (within the limitations inherent in FORTRAN, see chapter IV) and modular fashion. The subroutines range from 51 to 366 lines in length with the majority being under

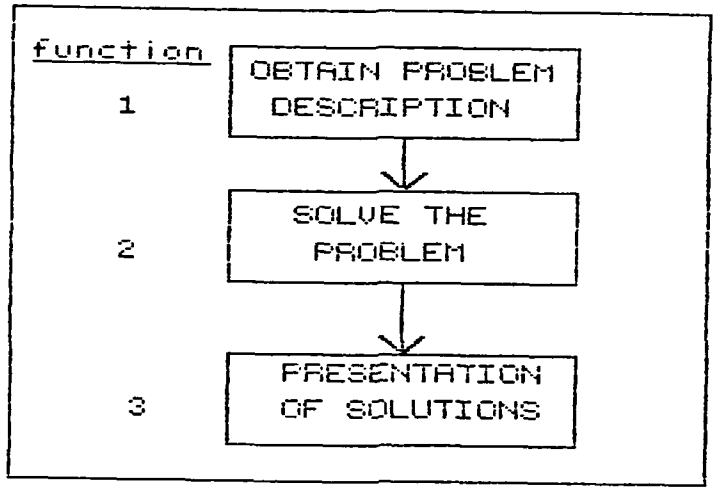


Figure 7: PCADA Functional Organization

125 lines.

Although a number of the program's subroutines serve as utilities that perform repetitious program functions (such as reading the user's input) at all levels of the system, the PCADA package is best described as consisting of three primary functions. The functional organization of the program is shown graphically in Figure 7.

Obtain Problem Description. The first program function is concerned with obtaining a problem description in the appropriate format in preparation for problem solution. The system offers two methods for obtaining a problem description. First, users may enter their problem as prompted by the PCADA System; and second, users may request the system to obtain the problem description from disk. New

problems must be entered into the system following the first method. Once entered though, they are automatically saved to disk for future reference.

Included in this function is a comprehensive capability to modify existing problem descriptions. The following modification options are available:

-- add/delete constraint

-- add/delete objective function

-- add/delete decision variable

-- edit constraint coefficients

-- edit constraint right hand side

-- edit objective coefficients.

As with new problem descriptions, problems are automatically saved to disk when they are modified.

Solve the Problem. The second program function is to solve the problem. Both the weighting and constraint techniques are offered. This function is designed to allow an iterative approach. That is, users who are unsure of solution ranges may wish to initially run the problem using the weighting technique over a large range of weights. This solution may indicate the need for a "finer" solution using a smaller weight increment. Instead, the user may wish to examine the results using one specific (user defined) set of weights. Alternatively, once the user determines the legitimate range that the objectives may assume by using the

weighting techniques he may wish to turn to the constraint technique to locate other than just corner-point solutions. The user determines the level of detail for both techniques by specifying the weight or constraint increments to any desired value.

In short, the user is free to thoroughly investigate the problem solution without being required to redefine the problem. Alternatively he is free to modify the problem following each iteration in order to examine solution impacts due to changing conditions.

Presentation of Solutions. The third and final program function is designed to present the problem solutions to users, as needed, in an easy-to-read format. Two types of solution displays are available. The first is simply a summary of the objective values corresponding to each non-dominated solution. The second is a detailed summary of each solution including objective values, objective weights (if applicable) and decision variable values. For all solution displays, the user is given the option of receiving results on the video display screen or in hardcopy form on the printer.

As with the problem solution function, the user is free to iterate on the solution presentation options. For example, users may first wish to review the objective summary on the display screen and then examine some or all of the solutions in detail using the screen, printer or both.

User Interface.

The program uses a friendly user-interface as described in chapter IV. The following user friendly features are included:

-- Online system documentation is available concerning any question or prompt simply by entering "HELP" or "?".

-- Presentation of a display of the current problem at any time simply by entering "@" (or "@P" to obtain a hardcopy printout of the problem).

-- All system outputs including prompts, menus, help messages, error messages, and solution displays are neat, easy to read and do not utilize "computer terminology".

-- System control of program outputs to prevent loss of data due to "scrolling".

-- Return to previous menu (i.e. error recovery) in most cases by simply hitting the return key.

-- Minimize user effort through usage of a menu directed rather than command directed approach.

-- Real time screening of user inputs to help prevent delayed and/or compounded errors.

Program usage is described in detail in the User's Manual (Attachment A) and program implementation is detailed in the Programmer's Manual (Attachment B).

Design Innovations

Even though microcomputer capabilities are rapidly improving and becoming increasingly sophisticated, the complexity of the multiobjective problem has demanded the usage of innovative techniques in order to achieve the

desired goals. Usage of traditional programming approaches in multiobjective program implementation, even with maximum and efficient use of the target system capabilities, would still leave the software unworkable or at best unresponsive. Successful microcomputer implementation of complex techniques depends on both technical expertise in knowing the computer capabilities, and design innovation.

For multiobjective decision techniques, the limited microprocessor speed and random access memory (RAM) capacity are major obstacles to successful implementation. As a result, a number of design tradeoffs were performed. These tradeoffs have resulted in the implementation of several innovative and non-traditional techniques.

Most design tradeoffs involve improving program response at the expense of program size, or vice versa. Since both of these resources (i.e. time and memory space) were at a premium in this implementation, the first design problem was to identify those areas where response time could be sacrificed, without detriment to the system, in order to gain valuable memory space. The additional memory space could then be applied to improving response time in other areas where such improvements are deemed necessary to insure user acceptance. The goal of this process is to match system resources to user requirements.

Following are descriptions of several of the major design trades and the resulting design innovations that were

performed during implementation of PCADA software package.

Handling of Menu/Help Data. The traditional approach used for outputting information from the program to the user involves the use of WRITE or PRINT statements. These statements reference program data areas that contain the actual characters that are to be output. These data areas are extremely space consuming. Each output character consumes exactly one byte. When the inplementation is space limited, the programmer may be tempted to eliminate some system oututs and abbreviate others. Both of these actions are undesirable in that they reduce the system's user friendliness.

For a user friendly multiobjective decision package a great number of system outputs are required. At a minimum, outputs must be descriptive and avoid the use of abbreviations. System outputs should also include a comprehensive HELP capability to insure user acceptance.

In PCADA over 50 menus and corresponding help displays are used. This represents over 20,000 characters of data. The traditional approach would then utilize 20,000 bytes of random access memory. Since PCADA required this space for other program capabilities, an analysis of alternative menu handling techniques was performed.

It was decided that all PCADA menus and help data would be stored on disk and loaded individually as needed. Instead of 20,000 bytes of RAM, 768 bytes is used for this

purpose. The cost of this space gain was an increase in response time since each new menu presentation would require a disk access.

A series of timing tests was run and it was determined that menus stored in RAM would result in an average of 1 second response time while disk menus require an average of 3.5 seconds of response time. This 2.5 second penalty was deemed to be an effective trade for the extra space since 3.5 seconds is well below the time needed for the user to read and assimilate the menu. However, it was decided that error messages should appear without delay, thus they are stored in RAM and generated in the more traditional way. Using this new menu generation approach, excess computer capacity in one area was redirected for use in other areas.

Centralized Input/Output Processing. Along the same lines, input and output program statements are among the most costly in terms of program size. Furthermore, since the need for input/output operations usually is widespread throughout most applications, these types of statements tend to be most common. It was decided that a set of input/output utilities would be implemented instead of using a READ or WRITE statement wherever needed. Now, when a program module needs to ask a question and get the user's response, it calls the appropriate input/output processor. System READs and WRITEs are isolated to one centralized location. This technique also prevents the proliferation of range and

error checks that should necessarily accompany all input/output operations for any user friendly system.

Although this technique has resulted in a considerable saving of random access memory space, it has introduced additional overhead in terms of time for input/output operations. Now instead of direct READ or WRITEs, input/output operations involve 4 or 5 extra subroutine calls. This added linkage increases execution time and accordingly penalizes response time. The penalty though was less than 50 milliseconds per response. Since menu to menu response was still below the user's normal reading/assimilation time, this time impact was considered to be a worthwhile trade.

Linear Programming Package. Unlike menu to menu response time, the time for the package to find the solution to the multiobjective problem was excessive and needed to be reduced to make the package successful. Initial estimates of the time needed for each linear programming iteration ranged from 10 to 30 seconds, depending on problem complexity. Although this time was not excessive, the cumulative time for problems requiring many iterations would be.

To reduce the execution time, the linear programming subroutine was redesigned. The new version employed larger memory work areas and in some cases redundant sections of code. The redundant code eliminated the overhead associated with establishing the linkage to use common code. The overall result was a larger subroutine, but a faster one,

now running from 2 to 9 seconds per iteration. Again, a programming technique had been used to direct the computer's resource where needed. In this case memory had been used to improve execution time.

Performance Data

Because system response time was considered to be a major factor in determining user acceptance, it was given close attention throughout the development process. Response time is defined as the time between the user's response to a system prompt and the next system prompt. Since PCADA runs on a dedicated personal computer, response times are fixed. They are not affected by other users and/or system loads as is the case of large scale computers.

All PCADA menus and questions are stored on disk and are retrieved in real time when needed. For a typical menu, ten lines in length, PCADA response times average 3.5 seconds. Since this is considerably less than the time needed by the user to assmilate the question, acceptable response time has been achieved. Response time is not significantly influenced by menu size.

All PCADA error messages are generated internally, and no disk access is required for their presentation. As a result, error messages are generated instantly.

Solution times are heavily influenced by the problem's complexity (i.e., number of objectives, number of decision variables, and number of constraints), as well as the

solution parameters (i.e., weight/constraint increment). For each problem iteration, PCADA requires from 2 to 9 seconds to compute a problem solution. Most problems will require less than 2 minutes to solve. For example, PCADA solved the example problem presented in chapter IV using the constraint technique with 7 steps in 28 seconds. However, a complex problem with 4 objectives using a weight increment of 0.05 requires almost 1.65 hours to complete. While running, PCADA keeps the user informed of its progress.

The immediate availability of the microcomputer based PCADA package is considered to be an acceptable tradeoff for the execution delays for complex problems. Overhead turnaround time associated with powerful large scale computers is often measured in days.

System Outputs

The main outputs provided by the PCADA system are the problem displays and the solution displays. Using the example given in chapter IV, Figure 8 contains the system generated problem description.

PCADA offers two different solution displays. The first is a summary of problem objective values. Figure 9 contains a sample of such a display. It was generated using the constraint technique on the example problem presented in chapter IV.

PROBLEM NAME = EXAMPLE

----- OBJECTIVES -----

1. MAXIMIZE BOMBER = +2.00x1 +1.00x2

2. MINIMIZE FIGHTER = +1.00x1 +2.00x2

----- SUBJECT TO THE FOLLOWING CONSTRAINTS -----

1. X1SUPPLY:	+1.00x1	+0.00x2	<=	10.00
2. X2SUPPLY:	+0.00x1	+1.00x2	<=	8.00
3. AC-LIMIT:	+1.00x1	+1.00x2	<=	12.00

Figure 8: Problem Description Display

SUMMARY OF OBJECTIVE VALUES FOR THE NON-DOMINATED SOLUTIONS

SOLUTION	OBJECTIVES ----- BOMBER	FIGHTER
1	22.0000	14.0000
2	21.0000	15.0000
3	20.0000	16.0000
4	19.0000	17.0000
5	18.0000	18.0000
6	17.0000	19.0000
7	16.0000	20.0000

Figure 9: Objective Values Summary Display

Using the same example, Figure 10 contains a sample of the detailed solution display available through the PCADA system. In this case, the 4th solution is shown.

```
NON-DOMINATED SOLUTIONS FOR PROBLEM: EXAMPLE

*** SOLUTION  4 ***

----- OBJECTIVES -----
1.  BOMBER   =      19.0000
2.  FIGHTER  =      17.0000

----- VARIABLES -----
Decision variable x1  =       7.0000
Decision variable x2  =       5.0000

----- CONSTRAINT SLACKS/SURPLUSES -----
Constraint # 1 ( X1SUPPLY ) had a SLACK   =      3.0000
Constraint # 2 ( X2SUPPLY ) had a SLACK   =      3.0000
```

Figure 10: Detailed Solution Display

All PCADA outputs are available on the video display screen or the on the printer in hardcopy form.

VI Conclusions and Recommendations

The PCADA package fulfills the thesis objective of developing a multiobjective decision analysis package for a personal computer. Its user friendly design makes the package useful for personnel of varied backgrounds in a problem solving environment. Although PCADA has undergone fairly extensive testing by the author and others, and seems to satisfy all of the research objectives, its true measure of success will be user acceptance over a period of time.

Nonetheless, all objectives and subobjectives as stated in chapter I have been implemented. The author is encouraged by this demonstration that today's personal computers can indeed be effectively utilized for implementation of valuable operations research techniques.

PCADA though represents only the "tip of the multiobjective iceberg". Other techniques including goal programming and the iterative weighting method (8:210) would be useful. In addition, helping decision makers establish their objective preferences would be a powerful addition to the package.

The constraint and weighting techniques as implemented could also be enhanced by adding sensitivity analysis. Finally, the area of result presentation should be investigated. The use of color and/or graphics for program outputs would greatly increase the program's effectiveness.

Bibliography

1. Ackoff, Russell L. "The Future of Operational Research is Past." The Journal of the Operational Research Society 30 (1979): 93-104.

2. Ackoff, Russell L. "Resurrecting the Future of Operational Research." The Journal of the Operational Research Society 30 (1979): 189-199.

3. Adelman, Leonard. "Real-Time Computer Support for Decision Analysis in a Group Setting: Another Class of Decision Support Systems." Interfaces 14 (1984): 75-83.

4. Amiry, Peter. "The Year of the Micro." The Journal of the Operational Research Society 32 (1981): 249.

5. Cassell, Douglas A. Microcomputers and Modern Control Engineering. Reston, Virginia: Reston Publishing Company, Inc., 1983.

6. Chankong, Vira, and Haimes, Yacov Y. Multiobjective Decision Making Theory and Methodology. New York: Elsevier Science Publishing Company, 1983.

7. Clementson, A. T., and Clewett, A. J. "Management, Operational Research and the Micro: A Critical Review of What is Available." The Journal of the Operational Research Society 32 (1981): 255-268.

8. Cohon, Jared L. and Marks, David H. "A Review and Evaluation of Multiobjective Programming Techniques." Water Resources Research 11 (1975): 208-220.

9. Cohon, Jared L. Multiobjective Programming and Planning. New York: Academic Press, 1978.

10. Conte, Captain Robert L. "Computer Assisted Analysis for Military Managers," Master's Thesis, Air Force Institute of Technology, 1979.

11. Douglas, A. S. "New Opportunities for O. R." The Journal of the Operational Research Society 32 (1981): 251-254.

12. Fraley, Major Theodore R. E., and Kem, Captain Dale A. "FORTRAN Based Linear Programming For Microcomputers," Master's Thesis, Air Force Institute of Technology, 1982.

13. Freedman, M. David, and Evans, Lansing B. Designing Systems with Microprocessors: A Systematic Approach. Englewood Cliffs, New Jersey: Prentice-Hall, Inc. 1983.

14. Fried, Louis. "Nine Principles For Ergonomic Software." Datamation November 1982: 163-166.

15. Gear, C. William. Computer Organization and Programming. New York: McGraw-Hill Book Company, 1969.

16. Geoffrion, A. M. "Can MS/OR Evolve Fast Enough?" Interfaces 13 (1983): 10-25.

17. Hammersley, P "The Impact of Microcomputer Systems on Commercial Data Processing." The Computer Journal 24 (1981): 14-16.

18. Hollocks, B. W. "Personal Computers and Operational Research - Experience in British Steel." The Journal of the Operational Research Society 32 (1981): 269-275.

19. Khandwalla, Pradip N. The Design of Organizations. New York: Harcourt Brace Jovanovich, Inc., 1977.

20. Lines, A. H. "The Microcomputer in Management." The Journal of the Operational Research Society 32 (1981): 297-301.

21. Lucas, Henry C. Jr. The Analysis, Design, and Implementation of Information Systems. New York: McGraw-Hill Book Company, 1976.

22. Microsoft. FORTRAN-80 Reference Manual. Bellevue, Washington: Microsoft, 1979.

23. Microsoft. FORTRAN-80 User's Manual. Bellevue, Washington: Microsoft, 1979.

24. Microsoft. Softcard Reference Manual. Bellevue, Washington: Microsoft, 1979.

25. Microsoft. Utility Software Manual. Bellevue, Washington: Microsoft, 1979.

26. Myers, Edith. "New Ways to Think Software." Datamation April 1982: 85-90.

27. Ranyard, J. C. "Introducing the Microcomputer Into the O. R. Department." The Journal of the Operational Research Society 32 (1981): 277-295.

28. Sanders, Donald H. Computers and Management In a Changing Society. New York: McGraw-Hill Book Company, 1974.

29. Shneiderman, Ben. "How To Design With the User in Mind." Datamation April 1982: 125-126.

30. Stevens, G. C. "O. R. Workers, Information Systems Analysts and the Challenge of the Micro." The Journal of the Operational Research Society 33 (1982): 921-929.

31. Turban, Efraim, and Meredith, Jack R. Fundamentals of Management Science. Dallas, Texas: Business Publications, Inc., 1977.

32. Wu, Nesa, and Coppins, Richard. Linear Programming and Extensions. New York: McGraw-Hill Book Company, 1981.

APPENDIX A:

PERSONAL COMPUTER AIDED DECISION ANALYSIS (PCADA)

USER'S MANUAL

CONTENTS

I INTRODUCTION 58

II PCADA CAPABILITIES AND LIMITATIONS 60

III INPUT RULES 63

Obtaining a Display/Printout of the Currently Active Problem 63
Obtaining HELP Concerning any PCADA Menu or Question 64
Responding to a Menu 64
Responding to a Question-Floating Point 65
Responding to a Question-Integer 66
Responding to a Question-Yes or No 66
Responding to a Question-Text 66

IV SYSTEM STARTUP 67

V SYSTEM USE 68

Problem Definition Function 71
Obtaining a Problem from Disk 71
Defining a New Problem 73
Entering Objective Functions 75
Entering Constraints 77
Editing a Problem 78
Add Objective Function 79
Delete Objective Function 80
Add Constraint 81
Delete Constraint 81
Add Variable 83
Delete Variable 83
EDIT Objective Coefficients 84
EDIT Constraint Coefficients 86
EDIT Constraint Right Hand Side 88
Problem Solving Function 89
Which Technique to Select 89
The Weighting Technique 90
The Constraint Technique 94
Presentation of Problem Results 96
No Bounded or Unbounded Solutions 97
Unbounded Solution Found 97
Non-Dominated Solution Set 98

I Introduction

The Personal Computer Aided Decision Analysis (PCADA) software package is designed to meet the needs of managers and analysts through easy-to-use, time responsive computer support. The package currently implements the weighting and constraint multiobjective decision analysis techniques. It is specifically designed to exploit the capabilities and responsiveness of commonly available desk-top computers.

The purpose of this document is to explain how to use PCADA. That is, how can PCADA be used by managers and analysts of all disciplines in solving multiobjective decision analysis problems.

Before proceeding, potential system users should do two things. First, the problem needs to be identified as a type suitable for solution by this package. Specifically, PCADA will solve multiple objective linear programming problems.

There are three characteristics that can be used to identify a multiobjective linear programming (MOLP) problem. First, the problem will have more than one objective. (Actually PCADA can also be used to solve single objective problems.) The second characteristic of MOLP problems is a set of one or more constraints associated with the objectives. Finally, all objectives and constraints must be linear functions. Before using PCADA to obtain the optimal problem solution(s) users should have their problem formulated and ready for input.

The second thing users should do before proceeding is to insure the availability of an appropriate personal computer. As implemented, PCADA requires a Z80 based microcomputer with 64K of random access memory, two disk drives, a printer, and a video display screen.

The remainder of this document describes how to use the PCADA software to solve MOLP problems. Section II gives program capabilities and limitations. Section III presents general program input/syntax rules. How the system is initiated is described in section IV. The detailed usage description is provided in section V.

II PCADA Capabilities and Limitations

The PCADA package is designed to solve multiobjective linear programming problems. It offers both the weighting and constraint techniques.

The weighting technique uses a linear combination of the objective functions to form a new objective function and then iteratively solves the new linear programming problem. There are a number of possible solutions because there are several linear combinations used as the new objective function. These solutions are candidates for the non-dominated solution set (NDSS). The constraint technique uses one objective function as the objective function of the new problem. The rest of the objective functions are adjoined as equality constraints with the right hand side equal to one of a range of values appropriate for the objective functions. Again the solutions to these problems are candidates for the NDSS. One characteristic of the weighted technique is that the only solutions found will be corner point solutions (intersections of constraints). The constraint technique on the other hand finds solutions all along the frontier defined by the constraints of the original problem.

When the user defines a MOLP problem for analysis, PCADA saves it to disk for future use. Thus it is not necessary to re-specify problem descriptions prior to each solution iteration. Furthermore, this facilitates sensitivity studies where users can examine the effect on answers of

subtle problem modifications. For this purpose, PCADA offers a broad range of modification options that also allow quick and easy re-solution as problem conditions change.

Following problem solution, PCADA offers extensive solution display options. At the users request, solution outputs can range from simple objective value summaries to detailed solution displays that contain objective values, objective weights (if requested, for the weighting technique only), decision variable values, and constraint slack or surplus values.

In general, PCADA is responsive with solutions requiring less than four seconds per iteration. However, as the number of objectives increases and the size of the iteration increment (as specified by the user) decreases, the number of iterations can quickly become large. Thus when establishing the solution parameters, users need to be aware that the time required to compute the set of non-dominated solutions is directly related. Some problems can require on the order of thirty or more minutes of uninterrupted computer time. For example, using the constraint technique, a problem with four objectives using ten steps to cover the range of each objective would require approximately 66 minutes to compute the non-dominated solution set. Still, this time compares vary favorably with the time that would be required using the normally less accessible "large-scale" computers (if available).

Because of the limited memory, disk, and speed capabilities of personal computers, the PCADA software can only process problems that are within the following limitations:

-- no more than 4 objectives

-- no more than 10 constraints

-- no more than 10 decision variables.

Problems that are more complex should be solved using large-scale computer based decision analysis packages.

III Input Rules

Users should respond to each PCADA prompt by entering the requested data on the computer keyboard. The return key must be pressed after the data value. Users need not be concerned with whether their inputs are upper case or lower case, as all lower case characters are internally converted to upper case.

All inputs are subjected to range and error checks by the software as they are made. When a problem is detected, the user is immediately informed with a descriptive error message. Error messages are generally self-explanatory and identify what is wrong with the input. Following the error message the prompt which was answered in error is again issued, giving the user the chance to respond correctly, and then continue normal program processing.

To simplify the input burden, a number of input rules apply throughout the PCADA system.

Obtaining a Display/Printout of the Currently Active Problem. During processing, users may need to refer to the problem being processed. To obtain a display of the problem on the video display screen users can enter "@" in response to any program menu or question. Figure 11 contains a sample of the problem display as generated by PCADA. Following problem display, program control reverts to the point where the request was made. Users desiring a permanent hardcopy display of their problem can enter "@P"

```
PROBLEM NAME = EXAMPLE

----- OBJECTIVES -----

 1. MAXIMIZE BOMBER  =     +2.00x1     +1.00x2

 2. MAXIMIZE FIGHTER =     +1.00x1     +2.00x2

-----SUBJECT TO THE FOLLOWING CONSTRAINTS -----

 1. X1SUPPLY:    +1.00x1     +0.00x2                    <=     10.00

 2. X2SUPPLY:    +0.00x1     +1.00x2                    <=      8.00

 3. AC-LIMIT:    +1.00x1     +1.00x2                    <=     12.00
```

Figure 11: Sample Problem Display

instead of "@". Again after the printing process is completed program control reverts to the point at which the request was made. NOTE: This option works only after a legitimate problem has been defined or loaded from disk.

Obtaining HELP concerning any PCADA menu or question. Although most PCADA prompts are self-explanatory, users that need assistance understanding any system menu or question can obtain clarifying information by entering "HELP" or "?" in response to the question. The system will respond to the request by presenting information directly related to the prompt on the video display screen. Following presentation of the HELP information, the original question or menu is redisplayed and processing continues in the normal fashion.

Responding to a Menu. The PCADA system makes extensive use of menus to determine a user's choice of processing

options. When presented with a menu, users should examine the options and enter the integer value corresponding to the desired option.

Floating point, text, integer values that exceed the menu's range, or other non-integer inputs will result in an appropriate error message. On most menus, an input of 0 (zero) will cause program control to revert to the previous menu. In this way, users can recover from erroneous menu selections. Finally, if the return key is entered with no input, PCADA assumes that the user wants the value 0 (zero). This feature can be used to save keystrokes when zero is desired or when the user wants to quickly revert to a previous menu.

Responding to a Question-Floating Point. PCADA asks several questions requiring a floating point value input. Users should respond by entering the value in decimal format. For example, the value 523 should be entered as "523." or "523.0" or simply "523". The system will reject other representations such as 5.23E+2. For these representations, text inputs, out of range values, or other illegal values the system will generate an error message as appropriate. Note that integer values are acceptable and are converted to floating point internally. In addition, as with menus, if the return key is entered with no input, a value of 0.0 is assumed. This feature can be used to quickly enter coefficients and constraint right hand sides whose value is zero.

Responding to a Question-Integer. PCADA asks several questions that require an integer valued response. (Menus are a special case of integer valued questions.) All inputs other than integers (including floating point values) will result in an appropriate error message. As before, if no input is entered, it is assumed to be the value 0 (zero).

Responding to a Question-Yes or No. The PCADA package asks several questions that require a YES or NO response from the user. All of these questions have been converted to menus. As indicated, users should enter "1" when wanting to answer "yes", and "2" when a "no" response is appropriate. All other responses will result in an explanatory error message.

Responding to a Question-Text. The PCADA package asks several questions that require a text valued response. The rules concerning text input vary depending on how the item is to be used. For example, the program will prompt the user for a name to be used to identify the problem. This item is required and is subsequently used as a disk file name for saving the problem. Because of its usage, the first character must be a letter (rather than a number or special character). In all cases, text rules are identified in the question and if needed, explanatory error messages are presented. For optional text inputs, entering the return key with no input will result in the default value of all blanks.

IV System Startup

Prior to initiating the PCADA software package, users should have an initialize diskette that is not write protected that can be used for storing the problems that are going to be worked. It is not necessary to use a different diskette each time, as each problem file requires less than 2K bytes of disk storage and many problems can be stored on each diskette.

If the user plans to access problem files that were created at a previous PCADA session, he should insure that he has the correct problem file names. All problem files are stored on the problem diskette using the name supplied by the user when the problem was first defined. Problem disk files can be identified as those of type "PRB".

When ready to begin after turning the power on to the computer system, the user should insert the write protected PCADA system diskette into disk drive #1 , and the problem file diskette (not write protected) into disk drive #2. At this point the PCADA software is started by entering the command: "PCADA." Detailed usage of the software from this point is described in the next section.

V System Use

The purpose of this section is to describe the use of all PCADA software functions. The chapter is organized into the three major program functions. In the first section those activities related to problem definition are described. The second section is devoted to the problem solution functions and the last section describes the presentation of problem results.

The problem definition function must be executed when the system is first started. However, once a problem has been defined or loaded from disk the user is free to alternate between the problem definition and problem solving functions as he tries different solution techniques and/or modifies his problem definition. The user is given the option of entering the examination of results function after each execution of the problem solving function.

When the user follows the instructions for system startup (see chapter IV), the program loads and the first system display is presented on the video display screen as follows:

```
**************************************************
*                                                *
*  PERSONAL COMPUTER AIDED DECISION ANALYSIS     *
*                                                *
*                    (PCADA)                     *
*                                                *
**************************************************
```

This program is designed to simplify the decision making process. It does so by providing an easy to use tool for solving multiple objective problems.

---- GENERAL INSTRUCTIONS ----

Although most program prompts are self-explanatory, you may receive clarifying information to any question by responding to it with "HELP" or "?". Once you have defined a problem or loaded one from disk, you may have it displayed on the screen (or at the printer) by entering "@" (for printer enter "@P") in response to any question or menu. Before continuing, please insure that you have an initialized diskette for problem files installed in disk drive 2.

WHEN READY TO BEGIN, PRESS THE CARRIAGE RETURN BUTTON (CR)

When the user is ready to proceed, he strikes the carriage return button (hereafter referred to as CR).

At this point, the screen will clear, and the user will be asked which means he wishes to use to obtain a problem definition. The following menu is used for this purpose:

Before you can solve a problem you must define a new one or retrieve a previously defined one from disk. Please indicate the means you wish to use to obtain an active problem definition: (NOTE: To exit the program now, enter "0")

1. Load a previously defined problem from disk
2. Define a new problem

The above menu is referred to as "system level menu #1". The description of each of the menu options is given in the next section.

Whenever the user completes a problem iteration, that

is, when he completes his examination of his problem solution results, he is again faced with a "system level" menu as follows:

```
Please select your next operation from the following menu:
     (NOTE:  To exit the program now, enter "0")

     1.  Retrieve a different problem from disk
     2.  Define a new problem
     3.  Edit the currently active problem
     4.  Solve the currently active problem
```

This menu is referred to as "system level menu #2". Options 1 and 2 cause the same processing sequence as options 1 and 2 of the previous menu. These options are described in the problem definition section. Option 3 allows the user to modify the currently active problem. It is also described in the problem definition section. The processing related to option 4 is decribed in the problem solving section.

If the user enters "0" to either of the system level menus, processing terminates and the following closing message is displayed:

```
                PERSONAL COMPUTER AIDED DECISION ANALYSIS
                             (PCADA)

     Problems that you defined this session have been saved to disk.  Your problem
     files can be identified on the problem diskette as those of type "PRB".  You
     should take care to manage those files by deleting or re-defining those that are
     obsolete.  Also, please make note of your problem file names for future use.

                          ---- END PROGRAM ----
```

Problem Definition Function

There are two methods available to users to obtain a problem definition for the PCADA system to solve. The first, as requested by selecting option 1 from either of the system level menus described above, loads a previously defined problem from disk. NOTE: Problems that are loaded in this fashion must have previously been defined directly by the PCADA system.

The second method for obtaining a problem definition is requested by selecting option 2 from either of the system level menus previously described. This method is used for new problems that are not already saved on disk. With this method users are prompted to enter all the information needed to complete a problem description.

Obtaining a Problem from Disk. This function will access the problem diskette that is installed in disk drive #2 to obtain the desired problem file. First though, the user must specify the name of the desired problem file. He is given the opportunity to do so by responding to the following question:

Please enter the name of the problem that you now want to load from disk:

NOTE: By definition, disk file names must be from 2 to 8 upper case alphanumeric characters with the first character being a letter. Special characters are not allowed.

The PCADA system appends the file type of "PRB" to each

problem file name but users need not enter the "PRB".

If the requested problem file is available on the problem disk and no read or other input/output error occurs, the problem is loaded into the program work areas and program control reverts to system level menu 2. That is, a problem has now been defined and the user is free to select the next program option.

If the requested problem file does not exist or some other input/output error occurs, the user is presented the following menu:

> A disk input/output error has occurred. Please make sure that the problem diskette is installed in drive #2, the disk drive door is closed, and that you have entered the correct file name.
>
> Would you like to try again?
>
> 1. Yes
> 2. No

As indicated, the program is unable to obtain the desired problem file for what could be one of many reasons. Even if none of the suggestions apply, it is a good idea to remove the problem diskette, then reinsert and "reseat" it and try again.

The spelling of the desired file name must match exactly. To obtain a list of file names of the problem diskette the user must exit the PCADA program and use the computer's disk operating system to examine the disk contents.

If all else fails, it is possible that somehow the problem diskette has been damaged. If this is the case, the

desired problem will have to be reentered and saved to a new diskette. For problems which are long or complex it is a good idea for users to keep backup copies of them. Refer to the computer's manuals for instructions on how to make back-up disk copies.

Defining a New Problem File. The PCADA system can be used to input new problems for analysis. This option is invoked by selecting option "2" from either of the system level menus previously discussed. When users first enter this function, the video display screen will be cleared and the following introductory message will appear:

```
                    DEFINE NEW MULTIOBJECTIVE PROBLEM
                    ---------------------------------

In this module, you will be defining a new multiobjective problem to be solved.
You should now be prepared to enter the objective functions (up to four), and
the constraints (up to 10).  NOTE:  If you make a mistake on some entry, you
can easily correct and/or modify your problem later using the EDIT module.
Also, your problem will automatically be saved to disk for later use.
```

The first required input for defining a new problem is the problem name. For this the user is prompted as follows:

```
Please enter the name you wish to use for referring to this problem:
    (NOTE:  If you wish to stop now, enter "STOP")
```

Again, filenames must be from 2 to 8 alphanumeric characters with the first character being a letter.

The system will prevent the inadvertent loss of existing problem files by checking to see if the name requested is already in use as a disk problem filename. If PCADA finds a conflict it asks the user if this was intended as

follows:

> The name you selected is already in use as a problem file name. Do you wish to redefine that problem now?
>
> 1. Yes
> 2. No

If the user selects option "1" (yes), then the existing file is lost and will be redefined with a new problem description. If the user selects option "2" (no), then the existing file is left intact and the user will be asked for a new problem name as before using menu 10 (see above).

Next, PCADA will ask the user how many decision variables the problem contains with the following question:

> How many variables does your problem contain?

NOTE: The number of decision variables in a problem can easily be changed later using the problem edit capabilities. Thus, if the user makes a mistake in input now, it can be corrected later.

Problems must contain at least one decision variable and are limited to a maximum of 10.

At this point, PCADA will enter a loop to obtain the problem's objectives. A maximum of four objectives is allowed and at least one is required. The program will continue to prompt the user for objective function data until the user has input four or until the user indicates that there are no more.

Entering Objective Functions. The first input related to objectives is the objective name. This item is not required for problem solutions but is available to help clarify problem results. Thus, the user should enter the name that you wish to associate with the objective. COST, PROFIT, or REVENUE would be typical of objective names. Abbreviations are encouraged since the name is limited to 8 alphanumeric characters. Since this item is optional, no input (i.e. CR only) will be accepted and processed as all blanks. Objective names are requested with the following menu:

> Please enter the name or target quantity that this objective represents. If there are no more objectives for this problem, enter "/".

NOTE: This menu is used to signal the PCADA system that there are no more problem objectives, by entering "/".

Next, the user specifies if the objective is to be maximized or minimized by responding to the following menu:

> Please specify the objective function type:
>
> 1. Maximize
> 2. Minimize

Now the user will be asked for the objective function coefficients. The prompts occur one at a time and there will be one for each decision variable in the problem. The menu will specify the identification of the decision variable as follows:

Enter objective coefficient for variable x1

Recall that entering CR with no input will be interpreted as a value of 0.0. This feature can be used to speed the objective input process. In addition, erroneous inputs can be corrected later using the PCADA edit functions. After the above question has been asked for each decision variable, the objective function has been completely specified. Control will revert back to the start of this section to receive function.

When the user indicates that there are no more objectives, the PCADA system will enter a similar loop to obtain the problem constraints. A maximum of 10 constraints are allowed and at least one is required. Users that plan to utilize the constraint technique are subjected to a different constraint limit. Since the constraint technique adds all but one of the objectives to the problem as constraints, the maximum number of constraints is related to the number of objectives as follows:

# of objectives	limit # of constraints
1	10
2	9
3	8
4	7

(applies to constraint technique only)

The program assumes the existence of the non-negativity constraints. Thus they need not be entered by the user.

Entering Constraints. The first input related to constraints is the constraint name. This optional item is offered to help clarify problem results. Users should enter the name that they wish to associate with the constraint. MONEY, LABOR, or MATERIAL would be typical of constraint names. Abbreviations are encouraged since the constraint name is limited to 8 alphanumeric characters. Since this item is optional, no input will be processed as all blanks. Constraint names are requested with the following menu:

> Please enter the name of the resource that this constraint represents. If there are no more constraints for the problem, enter "/".

NOTE: This menu is used to signal the PCADA system that there are no more problem constraints by entering "/".

Next, the user is prompted for the constraint coefficients. The values are requested one at a time, one for each decision variable in the problem. The question will specify the identification of the variable as follows:

> Enter constraint coefficient for variable x1

As before, blank (CR only) inputs will be interpreted as a value of 0.0 and, erroneous inputs can easily be corrected later using PCADA edit functions.

Next, the type of relation that is to be used for the constraint is requested. That is, is the resource that the constraint represents lower bounded, upper bounded, or to be exactly specified? The relation type is requested via the

following menu:

Please specify the type of relation applicable to this constraint:

1. less than or equal
2. equal
3. greater than or equal

Finally, the limiting value that pertains to the constraint must be specified by the user. PCADA software uses the following question to request this:

Please specify the right hand side value for this constraint:

As with objective/constraint coefficients, a blank input will be interpreted as the value 0.0. Also, if an error is made, constraint right hand side can be corrected later using PCADA edit capabilities.

Editing a Problem. The PCADA system offers users the opportunity to alter problem parameters without having to completely respecify the entire problem. This option can be used when users detect an error in problem specification, or if the problem conditions have changed. The editing function is invoked by selecting option "3" from system level menu #2. NOTE: System level menu #2 is generated automatically after each major processing request. For example, after defining a problem, control reverts to system level menu #2. This section describes the nine problem editing functions provided by the PCADA system.

When the user first indicates his desire to enter the

problem modification function, the screen is cleared and the following introductory remarks and menu are presented:

EDIT PROBLEM

This function allows you to EDIT the problem that is currently loaded into program memory. Whenever an objective, constraint, or variable is to be referenced by number, the numbers are as they appear on the problem printout that can be obtained by entering "@P" (or "@" for screen) in response to any menu.

Please select from the following EDIT functions:
(NOTE: Enter "0" to exit the EDIT function)

1. Add objective function
2. Delete objective function
3. Add constraint
4. Delete constraint
5. Add variable
6. Delete variable
7. EDIT objective coefficients
8. EDIT constraint coefficients
9. EDIT constraint right hand side

This menu is referred to as the main editing menu, from it the user indicates the editing option that is needed. Following completion of each editing task, the above display is again presented giving the user the opportunity to perform additional editing functions. As indicated, when the user has completed all editing operations he should enter "0" (or just the CR), to cause program control to revert to system level menu #2, from which the problem can be solved.

Add Objective Function. Before allowing the user to add an objective function PCADA will check to make sure that the problem currently has less than the program limit of four objectives. If the problem has 4 objectives, the following message is displayed:

**** YOUR PROBLEM ALREADY HAS THE MAXIMUM OF 4 OBJECTIVE FUNCTIONS ****

The program then reverts to the main editing menu that was presented at the start of this section to process the next editing request.

If the problem has less than 4 objectives, the program will prompt the user for all the necessary objective information. The process is as described above under the "Entering Objective Function" section. As with all editing functions, following completion of the action, the main editing menu is presented for additional editing requests.

Delete Objective Function. Before allowing the user to delete an objective function, PCADA checks to make sure that the current problem has at least two objectives. Each problem must have at least one objective and PCADA will not allow deletion of the last objective. If the user selects this option and the problem has only one objective, the following messages is displayed:

```
**** YOUR PROBLEM CURRENTLY HAS THE MINIMUM ALLOWABLE 1 OBJECTIVE ****
```

If the problem has more than one objective then objective deletion is allowed and the user will select the objective to be deleted from the following menu:

```
DELETE OBJECTIVE

    1. objective 1
    2. objective 2
    3. objective 3
    4. objective 4

Please specify which of the above objectives you now wish to delete:
    (NOTE: Enter "0" if you have changed your mind)
```

When executed the above menu will contain from two to four objectives depending on how many objectives the problem contains. In addition, rather than "*objective n*" the menu will contain the name that was assigned to the objective when the problem was defined.

The user should select the objective that is to be deleted. If the user decides that he now does not want to delete an objective, he should enter "0" to keep the problem unchanged.

Add Constraint. Before allowing the user to add a constraint to the problem, PCADA will check to make sure that the problem currently has less than the program limit of ten constraints. If the problem already has ten constraints, the following message is displayed:

**** YOUR PROBLEM ALREADY HAS THE MAXIMUM OF 10 CONSTRAINTS ****

The program then reverts to the main editing menu to process the next editing request.

If the problem has less then 10 constraints the program will prompt the user for all the necessary constraint information. The process is as described under the "entering constraint" section. As with all editing functions, following completion of the action, the main editing menu is presented for additional editing requests.

Delete Constraint. Before allowing the user to delete a constraint, PCADA checks to make sure that the

current problem has at least two constraints. Each problem must have at least one constraint and PCADA will not allow deletion of the last constraint. If the user selects this option and the problem has only one constraint, the following message is displayed:

```
**** YOUR PROBLEM CURRENTLY HAS THE MINIMUM ALLOWABLE 1 CONSTRAINT ****
```

If the problem has more than one constraint, then constraint deletion is allowed and the user will make his choice from the following menu:

```
DELETE CONSTRAINT

    1. constraint 1
    2. constraint 2
    3. constraint 3
            .
            .
    (up to 10 constraints)

Please specify which of the above constraints you now wish to delete:
    (NOTE:  Enter "0" if you have changed your mind)
```

When this option is executed, the above menu will contain from two to ten constraints depending on the current problem. In addition, rather than "*constraint n*", the menu will contain the name that was assigned to the constraint when the problem was defined.

The user should select the constraint that is to be deleted. If the user decides that he now does not want to delete an constraint, he should enter "0" to prevent the problem from being altered.

Add Variable. This function is used to add decision variables to the current problem. Before adding a variable, PCADA checks the current problem to insure that it is not already at the variable limit of ten. If it is, then the following message is displayed:

```
**** YOUR PROBLEM ALREADY HAS THE MAXIMUM OF 10 VARIABLES ****
```

If the problem contains less than 10 variables then a variable is added. The new variable is assigned the next sequential variable number. That is, if the problem currently contains six variables the new variable will be X7. All constraint and objective coefficients corresponding to the new variable are initialized to a value of 0.0. The appropriate EDIT functions should be used to set these coefficient values where appropriate (see below). When the variable is added, the following message is displayed:

```
VARIABLE ADDED = xN

A variable has been added to your problem. The coefficient values for this
variable for all objectives and constraints has been initialized to a value of
0.0. If you need to alter these values, you may do so using the EDIT functions
to EDIT objectives and constraints accordingly.

          PRESS THE RETURN KEY WHEN READY TO CONTINUE...
```

NOTE: N will be set to the added variable number, a value from 2 to 10.

Delete Variable. This function is used to delete decision variables from the current problem. Before deleting a variable, PCADA checks the current problem to

insure that it is not already at the variable minimum of 1. If it is, then the following message is displayed:

```
**** YOUR PROBLEM ALREADY HAS THE MINIMUM OF 1 VARIABLE ****
```

If the problem currently contains more than one variable then a variable may be deleted. To determine which one to delete, PCADA presents the following menu:

```
Please input the number of the variable you wish to delete:
    (NOTE:  Enter "0" if you have changed your mind)
```

As indicated above, if the user no longer wishes to delete a variable he can avoid doing so by entering "0". Otherwise, the user should specify which variable to delete.

It is important to note that when a variable is deleted, the remaining variables are uppacked. That is, if the current problem contains six variables and the user deletes variable X4, then X5 will be redefined as X4 and X6 will be redefined as X5.

Edit Objective Function Coefficients. This function is used to alter the objective coefficients. When selected, a menu is presented as follows:

```
EDIT OBJECTIVE FUNCTION

    1.  objective 1
    2.  objective 2
    3.  objective 3
    4.  objective 4

Please specify the number of the objective function whose coefficients you
would like to EDIT:  (NOTE:  Enter "0" if you are done editing)
```

The above menu will contain only the number of entries as there are objectives in the current problem. In addition, in the operational menu '*objective n*' is replaced by the objective name as specified when the problem was defined.

The user should select the objective that he wishes to EDIT. When no more objective functions are to be edited, the user should enter '0'. Control will be returned to the main problem editing menu. Once the objective has been selected, the specific coefficient that is to be modified needs to be identified. For this, the following menu is presented:

EDIT OBJECTIVE FUNCTION

Following are the objective coefficients for objective #*n*: *name*

1. Coefficient for variable x1 = *value*
2. Coefficient for variable x2 = *value*
3. Coefficient for variable x3 = *value*

.
.

(repeated for each decision variable)

Please specify the number of the coefficient that you would like to change:
(NOTE: When done editing, enter "0")

"*Objective #n*" will indicate the number of the selected objective, "*name*" will indicate that objective's name. In addition, the menu will contain one entry for each decision variable, to a maximum of 10. The user should select that coefficient he wishes to edit. When no more coefficients of this objective are to be changed, the user should enter "0". He will then be returned to the menu which gives him the

option of editing other objectives.

If the user decides to edit one of the coefficients the following menu is presented:

Please specify the new coefficient value:

The user should enter the new objective coefficient as a floating point value. After entering the value, control reverts to menu 28 to allow the user to edit the other coefficients in the objective.

Edit Constraint Coefficients. This function is used to alter the constraint coefficients. When selected, a menu is presented as follows (menu 30):

EDIT CONSTRAINT

1. *constraint 1*
2. *constraint 2*
3. *constraint 3*

.
.

(up to 10 constraints)

Please specify the number of the constraint whose coefficients you would like to EDIT: (NOTE: If you are done editing coefficients, enter "0")

The above menu will contain only the number of entries as there are constraints in the current problem. In addition, in the operational menu '*constraint n*' is replaced by the constraint name as specified when the problem was defined.

The user should select the constraint that he wishes to EDIT. When no more constraints are to be edited, the user should enter '0'. Control will be returned to the main

problem editing menu. Once the constraint has been selected, the specific coefficient that is to be modified needs to be identified. For this, the following menu is presented:

EDIT CONSTRAINT

Following are the constraint coefficients for constraint #*n*: *name*

1. Coefficient for variable x1 = *value*
2. Coefficient for variable x2 = *value*
3. Coefficient for variable x3 = *value*

.
.

(repeated for each decision variable)

Please specify the number of the coefficient that you would like to change:
(NOTE: When done editing, enter "0")

"*Constraint #n*" will indicate the number of the selected constraint, "*name*" will indicate that constraint's name. In addition, the menu will contain one entry for each decision variable, to a maximum of 10. The user should select that coefficient he wishes to edit. When no more coefficients of this constraint are to be changed, the user should enter "0". He will then be given the option of editing other objectives.

If the user decides to edit one of the coefficients the following menu is presented:

Please specify the new coefficient value:

The user should enter the new constraint coefficient as a floating point value. After entering the value the user is given the option to edit other constraint coefficients.

Edit Constraint Right Hand Side. This function is used to alter the constraint right hand side. When selected, a menu is presented as follows:

EDIT CONSTRAINT

1. *constraint 1*
2. *constraint 2*
3. *constraint 3*

.
.

(up to 10 constraints)

Please specify the number of the constraint whose right hand side you would like to EDIT: (NOTE: If you are done editing right hand sides, enter "0")

The above menu will contain only the number of entries as there are constraints in the current problem. In addition, in the operational menu '*constraint n*' is replaced by the constraint name as specified when the problem was defined.

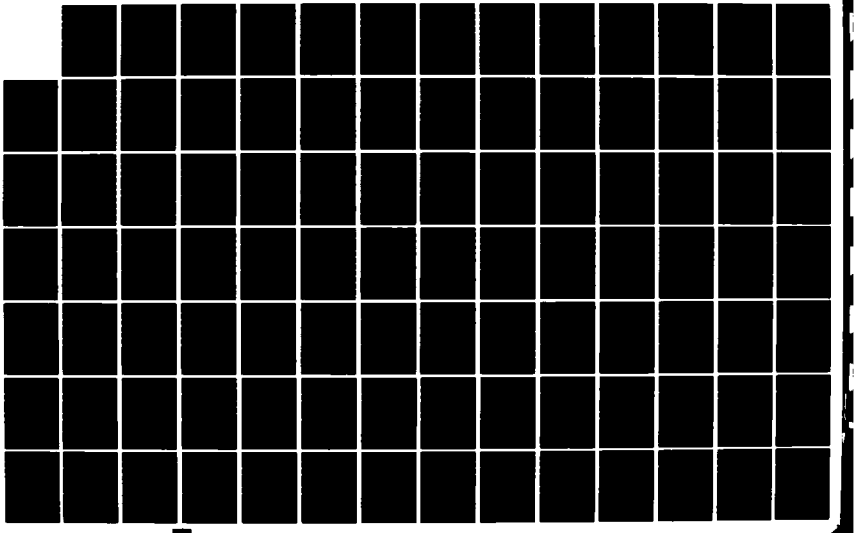
The user should select the constraint that he wants to EDIT. If no more constraints are desired, the user should enter '0'. Once the desired constraint has been selected, the new right hand side value needs to be specified. For this, the following menu is presented:

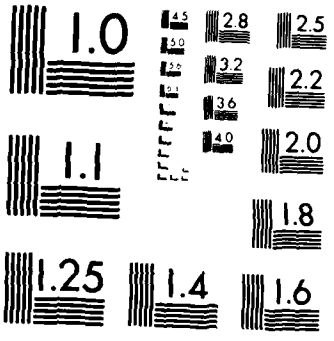
The right hand side for constraint #*n* is *value*

Please enter the new right hand side value:

The user should enter the new right hand side floating point value. After entering the value, the user will be given the oppurtunity to specify additional constraints that require a modification to the right hand side.

AD-A151 911 PERSONAL COMPUTER AIDED DECISION ANALYSIS(U) AIR FORCE 2/3
INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF
ENGINEERING G R WHITE 14 DEC 84 AFIT/GSO/OS/84D-8
UNCLASSIFIED F/G 9/2 NL





Problem Solving Function

The problem solving function is invoked by entering "4" in response to system level menu #2. Prior to selecting the problem solving function users should have completed all necessary problem modifications. Once invoked, the user should select from the two available problem solving methods. The options are presented in the following menu:

> Please specify which problem solving technique you now wish to use to solve the current problem (enter "0" to exit):
>
> 1. The weighting technique
> 2. The constraint technique

Which Technique to Select. First time users may be unsure whether they should select the weighting or the constraint technique. Since they both generate the non-dominated solution set, what's the difference? Simply stated, the weighting technique is faster but will generate only corner point solutions of the problem's feasible region. The constraint technique requires more computer time but can generate a more complete solution set. In addition, since the constraint technique optimizes only a single objective over a preset range of values for the other objectives, it is necessary to have some prior knowledge of the legitimate range of the other objectives.

Thus, a good procedure for solving new problems would be to first use the weighting technique to get an idea of the range of solutions that can be expected, and then use

the constraint technique to obtain a detailed solution set from the desired solution interval.

The Weighting Technique. The weighting technique will solve a series of single objective problems. In each problem, the aggregate objective is formed by multiplying each problem objective by a weight and summing them together. Each solution then represents the optimal allocation of resources for that specific set of objective weights. When solved over the complete range of weights for each objective, the solution set will reflect optimal solutions for the complete range of priorities.

The time required for PCADA to solve a problem with the weighting technique is a function of the number of iterations. PCADA uses only objective weight combinations that are "naturally normalized". That is, weight combinations that sum to one. The number of iterations is determined by number of objectives and the user selected weight increment as follows:

weight increment	-----number of objectives-----			
	1	2	3	4
1.000	1	2	3	4
0.500	1	3	6	10
0.333	1	4	10	20
0.250	1	5	15	35
0.200	1	6	21	56
0.100	1	11	66	286
0.050	1	21	231	621

For example, the 10 iterations that are required for 3 objectives with a weight increment of 0.333 would use the

following weight combinations:

Iteration	Objective weights 1	2	3
1	0.000	0.000	1.000
2	0.000	0.333	0.667
3	0.000	0.667	0.333
4	0.000	1.000	0.000
5	0.333	0.000	0.667
6	0.333	0.333	0.333
7	0.333	0.667	0.000
8	0.667	0.000	0.333
9	0.667	0.333	0.000
10	1.000	0.000	0.000

As can be seen, the number of iterations can easily become large. Depending on problem complexity (i.e. the number of decision variables and constraints), each iteration will require from 2 to 9 seconds of computer time. Thus some problems will require up to an hour or more to solve. Be patient, PCADA will tell the user how many iterations are required and will keep him informed of solution progress.

Once the user indicates that he wants to use the weighting technique, PCADA will present the following introductory message:

THE WEIGHTING TECHNIQUE

The weighting technique solves the problem by assigning weights to each of the separate objectives, forming a new objective, and optimizing it. As a result, the program will generate a "set" of solutions with each one corresponding to a different weight combination. These "non-dominated" solutions will be output and can be then compared to determine which one(s) is(are) appropriate.

Next the user will be presented with the following menu to indicate which weighting technique option is desired:

```
Please select from the following weighting technique solution options:
    (NOTE:  Enter "0" to exit the weighting technique)

    1.  Solve with specific objective weights
    2.  Solve with a range of weights on all objective functions
```

If the user wants to solve the problem once, using a specific set of weights, the user should select option 1. In this case, the program will next ask the user to input the desired weights for each objective:

```
Please specify the weight for objective #n: name
```

During program execution "*n*" will be replaced with the sequential objective number and "*objective number*" will be replaced by the respective objective name as defined by the user when the problem was defined. The program will accept any input from 0.0 to 1.0 for this prompt. It is not required that all of the input objective weights sum to 1.0.

At this point, the problem is solved using the weights as specified. After a 2 to 9 second delay, processing reverts to the Presentation of Problem Results function which is described below.

If instead of using the "solve with specific weights" option, the user would like to examine a more complete solution set covering the complete range of objective values, he should select the "range of weights" option.

When solving a problem with a range of weights, PCADA will automatically generate weight values which will cover

the complete range from 0.0 to 1.0 for all objectives. The user though must specify the desired weight increment. That is, how "fine" of a change in the aggregate objective is the user interested in. To determine the desired weight increment the PCADA system presents the following menu:

Please specify the weight increment to be used for this solution:
(NOTE: Enter "0" to exit the weighting technique)

1. 1.000	5. 0.200
2. 0.500	6. 0.100
3. 0.333	7. 0.500
4. 0.250	

NOTE: The smaller the selected increment, the more iterations and longer time will be required for the solution.

At this point, PCADA will attempt to solve the problem. Since the solution may require considerable time, the screen is cleared and the following display is presented:

---> MULTIOBJECTIVE PROBLEM SOLUTION IN PROCESS <---

PLEASE BE PATIENT!

As defined, this problem will require, *nnn* linear program iterations.

CURRENTLY WORKING ITERATION #:

*** *xxx* ***

The item "*nnn*" will specify how many iterations are needed and "*xxx*" will indicate which iteration is currently being worked. At the completion of all iterations program

processing reverts to the result presentation function.

The Constraint Technique. Like the weighting technique, the constraint technique will solve a series of single objective problems. Specifically, all but one of the objectives are added to the problem as equality constraints. The value of each objectives is changed for each iteration and the optimum value of the other objective is obtained. Thus each iteration finds the best solution given preset values for all but one of the objectives.

As with the weighting technique, the time required for PCADA to process the constraint technique is dependent on the number of iterations. The number of iterations is easily calculated as a function of the number of objectives and the objective range increment to be used as specified by the user. The number of iterations is calculated by raising the objective increment to the number of objectives - 1 power. For example a problem with 3 objectives and a user specified increment of 10 will require 10 x 10 = 100 iterations. The number of iterations is highly sensitive to objective increment.

Once the user has indicated his desire to use the constraint technique, PCADA will present the following introductory message:

THE CONSTRAINT TECHNIQUE

The constraint technique solves multiobjective problems by making all but one of the objectives into equality constraints and optimizing the one remaining objective. The right hand sides of the "new" constraints are bounded by the user as to the acceptable upper and lower limits. These right hand sides are then varied between the limits to obtain a set of acceptable solutions.

At this point, all but one of the objectives are added to the problem as constraints. PCADA retains the first objective as the one to be optimized. For each of the other objectives, the upper limit, lower limit, and range increment must be specified. First, PCADA will identify the objective to be optimized as follows:

> Objective #1: *name* will be optimized

Then, for each of the other objectives the following menus are presented:

> --- OBJECTIVE #*n*: *name* ---
>
> Please enter the upper acceptable limit for the above objective:

For this, the user should specify the highest objective value that he considers to be appropriate. Then:

> Please enter the lowest accepable limit for the objective function specified above:

As indicated, the lowest acceptable objective value should be input. Finally:

> Please specify the number of steps for solving the problem in going from the constraint lower limit to the upper limit:

At this point the user should specify the increment to be used in going from the lower to upper limit. The smaller the increment, the more program iterations and the longer the solution time.

Now, PCADA will attempt to solve the problem. Since the solution may require a considerable length of time, the screen is cleared and the following display is presented:

---> MULTIOBJECTIVE PROBLEM SOLUTION IN PROCESS <---

PLEASE BE PATIENT!

As defined, this problem will require, *nnn* linear program iterations.

CURRENTLY WORKING ITERATION #:

xxx

The item "*nnn*" will specify how many iterations are needed and "*xxx*" will indicate which iteration is currently being solved. At the completion of all iterations, program processing reverts to the result presentation function.

Presentation of Problem Results

This function is automatically invoked following each execution of the problem solving function. The purpose of this function is to allow the user to examine the problem solutions.

Before the user can examine the non-dominated solution

set, PCADA will inform the user of any special or unusual conditions that were encountered during the problem solution phase. A summary of these conditions follows.

No Bounded or Unbounded Solutions Found. If PCADA detects no bounded (that is, feasible) solutions or unbounded solutions, the user is so informed as follows:

```
----> Your problem had no feasible solutions <----

Check for error in formulation and/or erroneous constraints/objective functions

WHEN READY TO CONTINUE, PRESS THE RETURN KEY
```

As indicated, this condition probably means that the problem is incorrectly formulated. If it is correct, then with the constraints as stated there is no solution. That is, the constraints cannot be satisfied. Since there are no solutions to examine, as soon as the user acknowledges the above result by entering the return key, control will revert to system level menu 2 for problem refinement or other action.

Unbounded Solution Detected. In the event that PCADA detects an unbounded solution, the program will take one of two actions depending on the number of objectives. If there is only one objective, then the entire solution set consists of the single unbounded solution. The user is informed of this case as follows:

```
----> Your single objective problem is unbounded. <----

Check for error in formulation and/or erroneous constraints/objective functions.

WHEN READY TO CONTINUE, PRESS THE RETURN KEY
```

As indicated, this condition usually means that the problem is incorrectly formulated or incorrectly specified. For example, inadvertent omission of a constraint could easily explain the occurrance of an unbounded solution. Since there are no constraints to examine, as soon as the user acknowledges the above result by entering the return key, control will revert to system level menu #2 for problem correction or other action.

If there is more than one objective, then PCADA will inform the user that at least one unbounded solution was detected. It will also display the weights or constraint limits that were in effect when the first unbounded solution was found. To inform the user of this condition, the following display is generated:

> At least one unbounded solution was obtained. This suggests a potential problem formulation error. Please check your problem carefully. The unbounded solution first occurred under the following conditions:
>
> | | OBJECTIVE | WEIGHT/LIMIT |
> |---|---|---|
> | 1. | Objective #n: name | weight |
> | | . | . |
> | | . | . |
>
> (repeated for each objective)
>
> PRESS THE RETURN KEY WHEN READY TO CONTINUE...

When the user acknowledges the above result by entering the return key, result presentation processing continues in order to display all other detected solutions (if any).

Non-dominated Solution Set. Normally, following execution of either the weighting or constraint multiobjective

techniques, there will be a set of non-dominated solutions. This section describes how the user goes about viewing this solution set.

First, PCADA will inform the user of the scope of the solution set. That is, the number of entries in the solution set is displayed as follows:

> There were *n* NON-DOMINATED solutions found.

In the event that there were no feasible solutions found, program control will revert to system level menu 2 as before. PCADA can only handle up to 30 solutions. If more than 30 solutions are detected given the user specified solution parameters, the user is informed as follows:

> ----> NOTE: There were more than the program limit of 30 solutions <----
>
> The first 30 NON-DOMINATED solutions are presented in what follows. To get the rest, you will have to adjust your problem and solve it again. Refer to the users manual for further information.

The user will still be able to examine the first 30 solutions in the usual fashion but will have to redefine his solution parameters and resolve the problem to obtain the rest. For example, users may resolve the problem using the constraint technique and set the new objective points (i.e. lowest acceptable value) at the point where the first 30 solutions stop.

At this point, the user is presented with a menu which gives him the various solution display options as follows:

Please specify your choice for examining problem solutions:
(NOTE: Enter "0" to exit the solution display module)

1. Objective value summary -- all solutions
2. Detailed information -- one specific solution
3. Detailed information -- all solutions

A selection of option 1 will result in a display of just the objective values that result from each of the solutions. Figure 12 contains an example of this display for a two objective problem with 10 solutions. The user would use this display to quickly see the range of objective values he can expect and the nature of objective tradeoffs.

If the user selects option 2, he will be presented with the details concerning one solution (which the user selects). Detailed solutions include objective values, decision variable values, constraint slacks/surpluses and objective weights (if requested for the weighting technique

SUMMARY OF OBJECTIVE VALUES FOR THE NON-DOMINATED SOLUTIONS

SOLUTION	OBJECTIVES ----- BOMBER	FIGHTER
1	22.0000	14.0000
2	21.0000	15.0000
3	20.0000	16.0000
4	19.0000	17.0000
5	18.0000	18.0000
6	17.0000	19.0000
7	16.0000	20.0000

Figure 12: Objective Value Summary

```
NON-DOMINATED SOLUTIONS FOR PROBLEM: EXAMPLE

          *** SOLUTION 4 ***

          ----- OBJECTIVES -----
        1.  BOMBER   =      19.0000
        2.  FIGHTER  =      17.0000

          ----- VARIABLES -----
     Decision variable x1  =       7.0000
     Decision variable x2  =       5.0000

    ----- CONSTRAINT SLACKS/SURPLUSES -----
Constraint # 1 ( X1SUPPLY ) had a SLACK   =      3.0000
Constraint # 2 ( X2SUPPLY ) had a SLACK   =      3.0000
```

Figure 13: Detailed Solution Display

only). Figure 13 presents a sample of a detailed solution display. When the user selects option 2, PCADA will ask which solution is desired via the following menu:

```
Please enter the number of the solution that you would like to see:
```

Solution numbers are as specified in the objective summary display.

If the user selects display option 3, he will be presented with the details concerning all solutions. Solution displays are presented one at a time and are in exactly the same format as shown in Figure 13.

For both display options 2 and 3, if the solution was obtained via the weighting technique, the user is asked if he would like the display to include the objective weights that correspond to the solution. This is asked using the

following menu:

Would you like your solution outputs to include the objective weights that were used to obtain the corresponding solution?

1. Yes
2. No

If the user selects option 1 (yes) then the detailed solution displays will include the objective weights in parentheses next to the objective value. If the user selects option 2 (no) this item will be omitted from the display as in Figure 13.

For all displays, the system will ask the user where he would like the solution displays to be presented. The following menu is used for this purpose:

Please indicate the desired output device for the solutions:

1. Display screen
2. Printer

If the user selects option 2 (printer), the PCADA system presents the following instructional message:

>> TO PRINT THE SOLUTIONS <<

1. Turn on the printer
2. Set the printer "on-line"
3. Align the paper

*** WHEN READY TO CONTINUE, PRESS THE CARRIAGE RETURN ***

When the user responds by pressing the carriage return, printing will begin.

If the user requests results to go to the display

screen, the screen will clear and output will begin.

For both options (printer or display screen), as soon as solution presentation is completed, the user will be given the opportunity to continue solution examination.

APPENDIX B:

PERSONAL COMPUTER AIDED DECISION ANALYSIS (PCADA)

PROGRAMMER'S MANUAL

CONTENTS

I INTRODUCTION 107

Software Environment 107
Programming Conventions and Style 107
Document Outline 108

II DISK FILES 109

The Menu/Help File 109
Problem Files 112

III COMMON BLOCKS 115

The Problem Common Block 115
The Menu/Help Common Block 118
The Solution Common Block 121
Common Block Cross Reference 126

IV PROGRAM LINKAGE 127

Subroutine Trees 127
Subroutine Cross Reference 131

V PROGRAM MODULE DESCRIPTIONS 132

CDELIN -- (Code Line). 133
CONMVE -- (Move Constraint). 136
CSOLVE -- (Solve With Constraint Tech.). . . . 139
DSKPRB -- (Retrieve Problem from Disk) 143
EDTPRB -- (Edit Problem) 146
GETANS -- (Get Answer) 156
GETCON -- (Get Constraint) 162
GETFLT -- (Get Floating Point Value) 165
GETINT -- (Get Integer Value). 169
GETOBJ -- (Get Objective) 173
GETPRB -- (Get Problem from Disk) 176
GETTXT -- (Get Text) 180
LP -- (Linear Programming) 184
NEWPRB -- (Define New Problem) 190
PCADA -- (Main Program) 194
PRBSLV -- (Problem Solve) 202
PROANS -- (Process Answer) 208
PROHLP -- (Process Help) 215
PRSMEN -- (Present Menu) 218
PRTANS -- (Print Answer) 222
PRTPRB -- (Print Problem) 231

CONTENTS (cont.)

SAVPRB -- (Save Problem) 236
SOLVE -- (Solev Problem) 239
WSOLVE -- (Solve with Weighting Technique) . . 243

VI MENU/HELP FILE DATA 247

I Introduction

The purpose of this document is to describe the PCADA software package to the level of detail required by programmers and analysts who may wish to modify and/or expand the program's capabilities. PCADA was programmed in such a way to facilitate fault isolation/correction and program additions. That is, a structured consistent approach is used throughout.

Software Environment

PCADA was written in ANSI standard FORTRAN. Specifically, Microsoft's FORTRAN-80 on an Apple II+ computer equipped with a Z80 "softcard" was used for program development. However, no FORTRAN-80 or Apple unique features were utilized. In addition to FORTRAN-80, the program development process required the usage of CP/M, the CP/M editor, LINK-80, amd the CP/M Dynamic Debugging Tool (DDT).

Programming Conventions and Style

The twenty four program modules (one main program and 23 subroutines) that comprise the PCADA system were all coded using a consistent style. In addition, a number of helpful coding conventions were adopted for all program modules.

First, to facilitate program readability, all statement labels appear in numerical order. As an example of this feature's usefulness, when examining PCADA listings, ana-

lysts can quickly locate specific sections of code by locating the desired label in its correct numerical position.

Second, all format statements are labeled beginning with 900 and continuing sequentially from there. Also, all format statements are located at the end of each program module rather than after a corresponding read/write statement that uses the format.

Finally, most local variables are explicitly declared as to their type. This feature helps prevent errors that can occur when variables default to what may be an incorrect implicit type.

Document Outline

Chapter II describes the structure and content of disk files that are required by the PCADA system. Common blocks are examined in detail in Chapter III. Included is a common block cross reference map. Chapter IV presents information relating to program linkage. In addition to a subroutine tree, a subroutine cross reference map is presented. A description of each PCADA module, including module listings is provided in Chapter V. Finally, the actual menu and help data that is used throughout PCADA system is contained in Chapter VI.

II Disk Files

The PCADA system utilizes two different disk files. They are, the menu/help file and the user defined problem files. The structure and content of these files is described in this chapter.

It should be noted that for all PCADA disk files, numeric data is stored on disk in numeric format. It is not converted to character format for disk storage. This speeds disk access and reduces the required disk storage space. In the file descriptions which follow, where applicable, the FORTRAN item name that pertain to each disk file item is presented in paranthesis following the item description.

The Menu/Help File

To help accomplish the goal of a system that is user friendly, detailed and lengthy menu and help information were determined to be essential. Because text type information can not be compressed, that is, each non-blank character requires one byte of storage and because the development system was limited to 64K of memory, it was decided that all menu and help information would have to be loaded from disk as needed. For this purpose, the menu/help file was created. Whenever a menu is to be generated, the system loads all of that menu's text and help data from the menu/help file on disk into the menu/help common block.

The menu/help disk file is called MENDAT.DAT and is

located on the system disk that is expected to be inserted into disk drive #1. When PCADA processing begins, the main program (PCADA) assigns logical unit #7 to the menu/help file. No PCADA software writes to the menu/help file and only one routine (PRSMEN) reads from it.

A complete listing of the contents of the menu/help file is presented in Chapter VI. The remainder of this section describes the structure of the menu/help file.

CP/M disk files consist of records that are each 128 bytes in length. In order to preserve disk space, rather than use "screen sized" 80 character records, trailing blanks are not stored and sequential menu lines are concatenated on the disk file. Menu size and format information is also stored to provide the software with the means to regenerate each menu when needed.

The menu/help file can be thought of as 51 separate blocks plus one control record. The 51 blocks correspond to the 51 menus that are used in the PCADA system. The size of each of these blocks is variable and can range from three to seven records (each record is 128 bytes) depending on the size of the corresponding menu. The control record is the first record in the file. This record is not used by the PCADA operational software. It does provide a space for keeping track of file version number, date of modification, and/or number of active records.

Following the control record, the separate menu blocks

appear sequentially. The first record of each menu block contains that menu's control information. Following is a description of this menu control information.

BYTE 1: The number of records containing menu data for this menu. Menu data records immediately follow the menu control record. (MENSIZ)

BYTE 2: The number of records containing help data for this menu. Help data records immediately follow the menu data records. (HLPSIZ)

BYTE 3: The number of lines that are required by the menu when output to the video display screen. This item may assume any value from one to twelve. (MENLIN)

BYTE 4: The number of lines that are required by the menu's help display when output to the video display screen. This item may assume any value from one to twelve. (HLPLIN)

BYTES 5-16: These 12 bytes are actually 12 separate items each of which corresponds to one line of menu data. If the menu is not twelve lines long then not all twelve items are used. For example, if MENLIN = 3, then only bytes 5, 6, and 7 of the 12 are used. Each byte contains the length, in characters, of the corresponding menu line. For example, if byte 5 = 69, then the first line of the menu has 69 characters in it. (MLNSZ)

BYTES 17-28: In a similar fashion to the above, these 12 bytes contain each of the individual line sizes for each of the up to 12 lines of help data that corresponds to the menu. (HLNSZ)

BYTES 29-128: Unused.

Following the control record described above, the actual menu and help text data appears. This data uses a minimum of two data records and may use up to six. The six record maximum is determined by the size of the common block item that is used to contain menu/help data when needed.

The item, MENVAL in common block MENDAT is currently sized to accomodate 768 characters.

Problem Files

Because multiobjective problems are generally lengthy and time consuming for users to enter into the program, it was decided that problem data would be automatically saved on disk. When storing or retrieving problem files, the PCADA system assigns logical unit #8 to disk drive #2. The system uses the user supplied problem name concatenated with the file type "PRB" as the problem file name.

In the PCADA system, access to the problem files is localized to just two subroutines. GETPRB is called when an existing problem is to be retrieved and SAVPRB is called to store the problem to disk following definition or modification. All problem data that is stored to the problem file on disk is contained in common block PROB during program execution.

Each problem file consists of three separate sections of information. The first section contains general problem description data. This data is contained on the first record in the problem file. The following information is included:

BYTE 1: The number of constraints in the problem. Due to program limitations, this value is a number from 1 to 10. (NCON)

BYTE 2: The number of objectives in the problem. Due to program limitations, this item will assume a value

between 1 and 4. (NOBJ)

BYTE 3: The number of variables in the problem. Due to program limitations, this item will assume a value between 1 and 10. (NVAR)

BYTES 4-128. Unused.

The second section of information contains data pertaining to the problem's objectives. This section will consist of up to four data records. The actual number of objective data records is specified by NOBJ. Each objective data record contains the following information:

BYTES 1-8: This field contains the user assigned objective name. Names are limited to eight characters. (ONAME)

BYTE 9: This field specifies the type of objective that this data record pertains to. Types are determined as follows: (OTYPE)
1 = MAXIMIZE
2 = MINIMIZE

BYTES 10-49: These 40 bytes contain the objective function coefficients. Each coefficient requires 4 bytes, thus space for 10 coefficients is allocated. This limit corresponds to the program limit of 10 decision variables. For each problem, NVAR specifies the actual number of coefficents that are used. Coefficient data is stored sequentially. For example, bytes 10 through 13 contain the coefficient for variable x1. (OBJ)

BYTES 50-128: Unused.

The final section of problem information is the constraint data. This section follows the objective section and consists of one data record for each constraint. There can be up to 10 constraint data records, but the actual number will be specified by NCON. Each constraint

data record contains the follwing information fields:

BYTES 1-8: This field contains the user assigned constraint name. Names are limited to eight characters. (CNAME)

BYTE 9: This field specifies the type of constraint that this data record pertains to. Types are determined as follows: (CTYPE)

1 = less than or equal
2 = equal
3 = greater than or equal

BYTES 10-49: These 40 bytes contain the constraint coefficients. Each coefficient requires 4 bytes, thus space for 10 coefficients is allocated. This limit corresponds to the program limit of 10 decision variables. For each problem, NVAR specifies the actual number of coefficents that are used. Coefficient data is stored sequentially. For example, bytes 10 through 13 contain the coefficient for variable x1. (CON)

BYTES 50-53: This field contains the constraint right hand side value. (RHS)

BYTES 54-128. Unused.

III Common Blocks

The PCADA software package uses three common blocks for the storage of problem, menu, and solution data. This chapter presents a description of these common blocks and a cross reference map showing which subroutines use which common blocks.

The Problem Common Block

Since many program modules in the PCADA system use problem data to accomplish their tasks, a common block was created to contain this data and facilitate the communication of it.

COMMON BLOCK NAME: PROB

COMMON BLOCK SIZE: 737 Bytes

Following is a description of the items contained in common block PROB.

ITEM NAME: CON

ITEM TYPE: REAL ARRAY; 10 by 10 elements

ITEM DESCRIPTION: CON contains the constraint coefficients for the problem. The 10 by 10 size reflects the fact that the program is limited to 10 constraints and 10 decision variables. The PCADA system limits the values that CON can assume to the range -99999.0 to +99999.0.

ITEM NAME: CNAME

ITEM TYPE: REAL*8 ARRAY; 10 elements

ITEM DESCRIPTION: CNAME contains the constraint names. The 10 elements reflect the fact that problems are limited to 10 constraints. Constraint names are optional, and can be from one to eight alphanumeric characters in length.

ITEM NAME: CTYPE

ITEM TYPE: INTEGER*1 ARRAY; 10 elements

ITEM DESCRIPTION: CTYPE contains the constraint types. There are three different types allowed and they are encoded in CTYPE as follows:

1 = less than or equal
2 = equal
3 = greater than or equal

The 10 element array size relects the fact that problems are limited to 10 constraints.

ITEM NAME: RHS

ITEM TYPE: REAL ARRAY; 10 elements

ITEM DESCRIPTION: RHS contains the constraint right hand side values. The 10 element size reflects the fact that problems are limited to 10 constraints. Constraint right hand side values are must be non-negative numbers less than 99999.0.

ITEM NAME: OBJ

ITEM TYPE: REAL ARRAY; 4 by 10 elements

ITEM DESCRIPTION: OBJ contains the objective coefficients for the problem. The 4 by 10 size reflects the fact that the program is limited to 4 objectives and 10 decision variables. The PCADA system limits the values that OBJ can assume to the range -99999.0 to +99999.0.

ITEM NAME: ONAME

ITEM TYPE: REAL*8 ARRAY; 4 elements

ITEM DESCRIPTION: ONAME contains the objective names. The 4 elements reflect the fact that problems are limited to 4 objectives. Objective names are optional, and may be from one to eight alphanumeric characters in length.

ITEM NAME: OTYPE

ITEM TYPE: INTEGER*1 ARRAY; 4 elements

ITEM DESCRIPTION: OTYPE contains the objective types. There are two different types allowed and they are encoded in OTYPE as follows:

1 = MAXIMIZE
2 = MINIMIZE

The 4 element array size relects the fact that problems are limited to 4 objectives.

ITEM NAME: NCON

ITEM TYPE: INTEGER*1

ITEM DESCRIPTION: This item contains the number of constraints that are currently part of the problem description. Because of system limitations, this item will assume values only between one and ten.

ITEM NAME: NOBJ

ITEM TYPE: INTEGER*1

ITEM DESCRIPTION: This item contains the number of objectives that are currently part of the problem description. Because of system limitations, this item will assume values only between one and four.

ITEM NAME: NVAR

ITEM TYPE: INTEGER*1

ITEM DESCRIPTION: This item contains the number of variables that are currently part of the problem description. Because of system limitations, this item will assume values only between one and ten.

ITEM NAME: FNAME

ITEM TYPE: REAL*8

ITEM DESCRIPTION: This item contains the name that was assigned by the user to the current problem. This item is also used to form the problem file name. That is, to FNAME, PCADA concatenates the file type ".PRB" before the problem is saved to disk. FNAME is a required input and can be from 2 to 8 alphanumeric characters in length. The first character must be a letter.

The Menu/help Common Block

This common block is used to contain menu and help data for the current menu. Whenever a new menu is required, the subroutine PRSMEN loads the menu data into the common block where it remains active until a different menu is needed.

COMMON BLOCK NAME: MENDAT

COMMON BLOCK SIZE: 798 Bytes

Following is a description of the items contained in common block MENDAT.

ITEM NAME: MENSIZ

ITEM TYPE: INTEGER*1

ITEM DESCRIPTION: This item contains the number of records that the current menu occupies on disk. Further details concerning this item can be found in the previous chapter which describes the menu/help disk file.

ITEM NAME: MENLIN

ITEM TYPE: INTEGER*1

ITEM DESCRIPTION: This item contains the number of lines that the current menu occupies on the video display screen. Further details concerning this item can be found in the previous chapter which describes the menu/help disk file.

ITEM NAME: HLPSIZ

ITEM TYPE: INTEGER*1

ITEM DESCRIPTION: This item contains the number of records that the help data which corresponds to the current menu occupies on disk. Further details concerning this item can be found in the previous chapter which describes the menu/help disk file.

ITEM NAME: HLPLIN

ITEM TYPE: INTEGER*1

ITEM DESCRIPTION: This item contains the number of records that the help data that corresponds to the current menu occupies on the video display screen. Further details concerning this item can be found in the previous chapter which describes the menu/help disk file.

ITEM NAME: MLNSZ

ITEM TYPE: INTEGER*1 ARRAY; 12 elements

ITEM DESCRIPTION: MLNSZ contains the size, in characters that each of lines in the current menu contains. The item is used to unpack the menu data in MENVAL for display. The 12 elements reflect the fact that PCADA limits the size of menus to 12 lines.

ITEM NAME: HLNSZ

ITEM TYPE: INTEGER*1 ARRAY; 12 elements

ITEM DESCRIPTION: HLNSZ contains the size, in characters that each of lines in the help data that corresponds to the current menu contains. The item is used to unpack the help data in MENVAL for display. As with menus, help displays are limited to 12 lines.

ITEM NAME: MENVAL

ITEM TYPE: LOGICAL ARRAY; 768 elements

ITEM DESCRIPTION: MENVALis used as a holding area for the actual menu/help character data that is presented to the user on the video display screen. It is loaded by PRSMEN directly from the menu/help file on disk. The 768 size was selected to accomodate the data from exactly six disk records (128 characters each). The data is packed, and must be interpreted by PRSMEN. The first MENSIZ times 128 characters contains the menu data and the next HLPSIZ times 128 characters is the help data. Trailing blanks are not stored.

ITEM NAME: ACTMEN

ITEM TYPE: INTEGER

ITEM DESCRIPTION: This item contains the identification of the currently active menu. Menus are identified by the starting disk record number. A complete listing of PCADA menus and help information is provided in Chapter VI. Included are the menu starting record numbers.

The Solution Common Block

Data pertaining to the problem solution is stored in a common block that is available to the many PCADA subroutines requiring this data. The data in this common block is set by various functions and used as needed. It is not saved to disk.

COMMON BLOCK NAME: SOLN

COMMON BLOCK SIZE: 3953 Bytes

The items contained in common block SOLN are described in the following:

ITEM NAME: W

ITEM TYPE: REAL ARRAY; 4 elements

ITEM DESCRIPTION: This item contains the objective weights or constraint limits that are to be used for the current solution iteration. If the weighting technique is being used, then weights are stored in W, otherwise constraints are stored. W is set iteratively as the problem is solved. Each element of this array corresponds to one of the objectives.

ITEM NAME: UB

ITEM TYPE: REAL ARRAY: 4 elements

ITEM DESCRIPTION: This item contains the objective upper limit to be used in solving the problem. For the weighting technique it is set to 1.0, for the constraint technique it is set to the user specified objective upper limit. When the problem is solved, W assumes values between LB and UB using the increment as specified in DEL.

ITEM NAME: LB

ITEM TYPE: REAL ARRAY: 4 elements

ITEM DESCRIPTION: This item contains the objective lower limit to be used in solving the problem. For the weighting technique it is set to 0.0, for the constraint technique it is set to the user specified objective lower limit. When the problem is solved, W assumes values between LB and UB using the increment as specified in DEL.

ITEM NAME: DEL

ITEM TYPE: REAL ARRAY: 4 elements

ITEM DESCRIPTION: This item contains the objective increment value to be used in solving the problem. For both techniques, this item is set to the increment as specified by the user. When the problem is solved, W assumes values between LB and UB using the increment as specified in DEL.

ITEM NAME: ISTRT

ITEM TYPE: INTEGER

ITEM DESCRIPTION: This item specifies which solution technique has been requested by the user. It is used as follows:

1 = the weighting technique
2 = the constraint technique

ITEM NAME: P

ITEM TYPE: REAL ARRAY; 30 elements

ITEM DESCRIPTION: This item contains the objective coefficients as used by the linear programming subroutine. The 30 elements are required for working storage by the LP subroutine. For the weighting technique, P contains the aggregate objective coefficients. For the constraint technique, P contains the objective coefficients of the first objective.

ITEM NAME: D

ITEM TYPE: REAL ARRAY; 10 by 31 elements

ITEM DESCRIPTION: This item contains the constraint coefficients as used by the linear programming subroutine. The 10 by 31 size is indicative of the 10 constraint limit and the working storage required by the LP subroutine.

ITEM NAME: IBV

ITEM TYPE: INTEGER ARRAY; 10 elements

ITEM DESCRIPTION: IBV is returned from the linear program subroutine containing the identity of the basic variables that are in the solution. For example, if IBV(1) is returned equal to 4, then x4 is in the solution.

ITEM NAME: SOLWHT

ITEM TYPE: REAL ARRAY; 30 by 4 elements

ITEM DESCRIPTION: This item contains the objective weights that corresponds to each non-dominated solution that is found. This item is used only for the weighting technique. The 30 dimension corresponds to the fact that PCADA will only process the first 30 solutions.

ITEM NAME: VARNUM

ITEM TYPE: INTEGER*1 ARRAY; 30 by 10 elements

ITEM DESCRIPTION: This item contains the values that are returned from the linear programming subroutine in IBV. Whenever the solution is determined to be of the non-dominated variety, PCADA copies IBV to VARNUM in order to preserve the solution for further processing. The 30 dimension corresponds to the fact the PCADA will retain only the first 30 solutions.

ITEM NAME: VARVAL

ITEM TYPE: REAL ARRAY; 30 by 10 elements

ITEM DESCRIPTION: This item contains the basic variable values that are returned from the linear programming subroutine. Whenever the solution is determined to be of the non-dominated variety, PCADA copies the solution to VARVAL in order to preserve the solution for further processing. The 30 dimension corresponds to the fact the PCADA will retain only the first 30 solutions.

ITEM NAME: OBJVAL

ITEM TYPE: REAL ARRAY; 30 by 4 elements

ITEM DESCRIPTION: This item contains the objective values that are computed by PCADA following each linear program iteration whenever the solution is determined to be of the non-dominated variety, PCADA copies the objective values to OBJVAL in order to preserve the solution for further processing. The 30 dimension corresponds to the fact the PCADA will retain only the first 30 solutions.

ITEM NAME: ZF

ITEM TYPE: REAL ARRAY; 4 elements

ITEM DESCRIPTION: This 4 element array is used as a temporary storage area for objective values after each linear program iteration. When the solution is determined to be a non-dominated solution, the values in ZF are moved to OBJVAL.

ITEM NAME: NNDS

ITEM TYPE: INTEGER

ITEM DESCRIPTION: This item contains the number of non-dominated solutions that have been found. It is incremented each time a new solution is found.

ITEN NAME: CNXREF

ITEM TYPE: INTEGER*1 ARRAY; 10 elements

ITEM DESCRIPTION: This item contains cross reference information that relates the constraint data in D to the constraint data in CON (in common block PROB). The two arrays are not kept parallel because LP requires the constraints to be presented in a specific order. If CNXREF(1) equals 3, it means that D(1) corresponds to the constraint stored in CON(3).

ITEM NAME: SPCFLG

ITEM TYPE: INTEGER*1

ITEM DESCRIPTION: This item is used to indicate the occurance of a special condition. A value of zero is default, -1 indicates that at least one infeasible solution was detected, and -2 means that at least one unbounded solution was detected.

ITEM NAME: UBWHT

ITEM TYPE: REAL ARRAY; 4 elements

ITEM DESCRIPTION: If an unbounded solution is detected, PCADA will inform the user of the constraint/weight conditions that were in effect when the solution was found. This array is used to save the values in effect for the first unbounded solution.

ITEM NAME: ICAP

ITEM TYPE: INTEGER

ITEM DESCRIPTION: This item is set to a non-zero value whenever more than 30 non-dominated solutions to a problem are detected.

Common Block Cross Reference

The following table indicates common block usage within PCADA by program module. The modules are listed in alphabetical order.

SUBROUTINE NAME	--- COMMON BLOCK NAME --- PROB	MENDAT	SOLN
CDELIN	X		
CONMVE	X		X
CSOLVE	X		X
DSKPRB	X		
GETANS	X		
EDTPRB	X		
GETCON	X		
GETFLT			
GETINT			
GETOBJ	X		
GETPRB	X		
GETTXT			
LP			X
NEWPRB	X		
PCADA	X	X	X
PRBSLV	X		X
PROANS	X		X
PROHLP			
PRSMEN		X	
PRTANS	X		X
PRTPRB	X		
SAVPRB	X		
SOLVE	X		X
WSOLVE	X		X

IV Program Linkage

This chapter presents information concerning the program linkage for the PCADA system. First a subroutine tree for the PCADA software package is presented. The tree appears in Figure 14 through 17. Next, a subroutine cross reference map is provided in Figure 18. This map is useful to programmers when it is necessary to determine the scope of proposed system changes.

Subroutine Trees

```
PCADA-----PRSMEN
          -GETTXT----PRSMEN
                     -PROHLP----PRTPRB----CDELIN
                     -GETANS    -PRSMEN
          -GETINT----PRSMEN
                     -PROHLP----PRTPRB----CDELIN
                     -GETANS    -PRSMEN
          -DSKPRB----GETTXT----PRSMEN
                               -PROHLP----PRTPRB----CDELIN
                               -GETANS    -PRSMEN
                      GETINT---PRSMEN
                               PROHLP----PRTPRB----CDELIN
                               -GETANS    -PRSMEN
                      GETPRB---OPEN
          -NEWPRB----(see FIGURE 15)
          -EDTPRB----(see FIGURE 16)
          -PRBSLV----(see FIGURE 17)
```

Figure 14: Top level Subroutine Tree

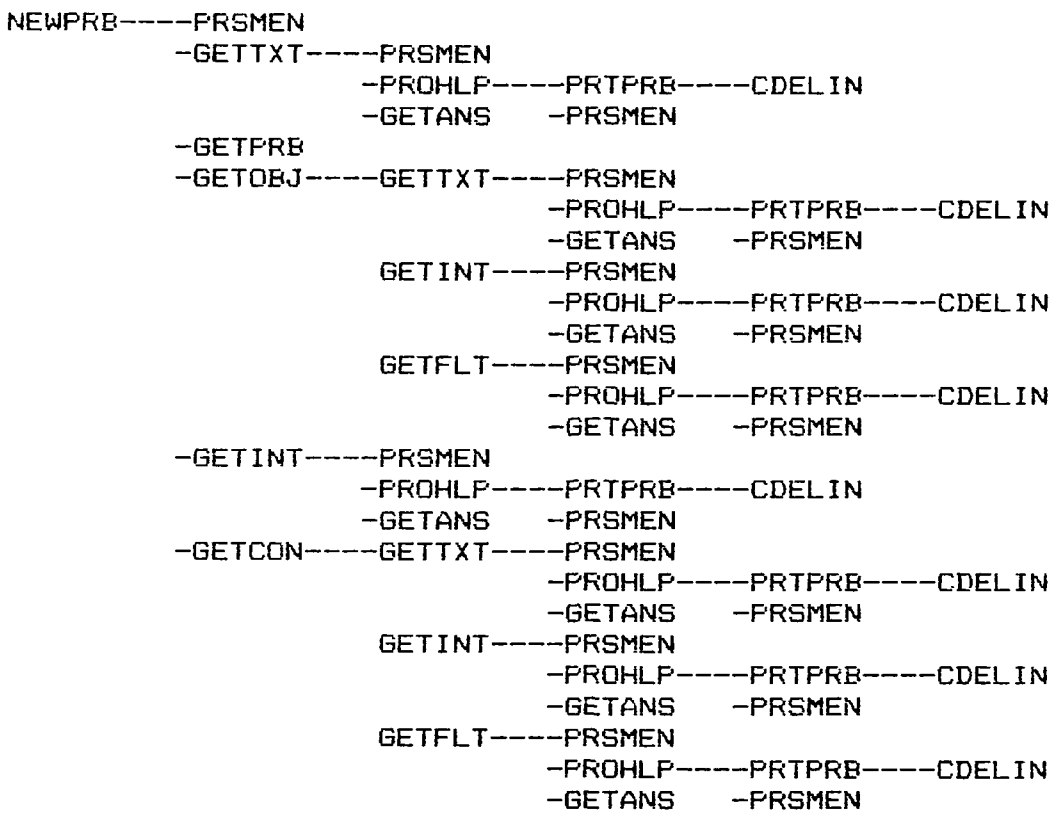


Figure 15: Define New Problem Subroutine Tree

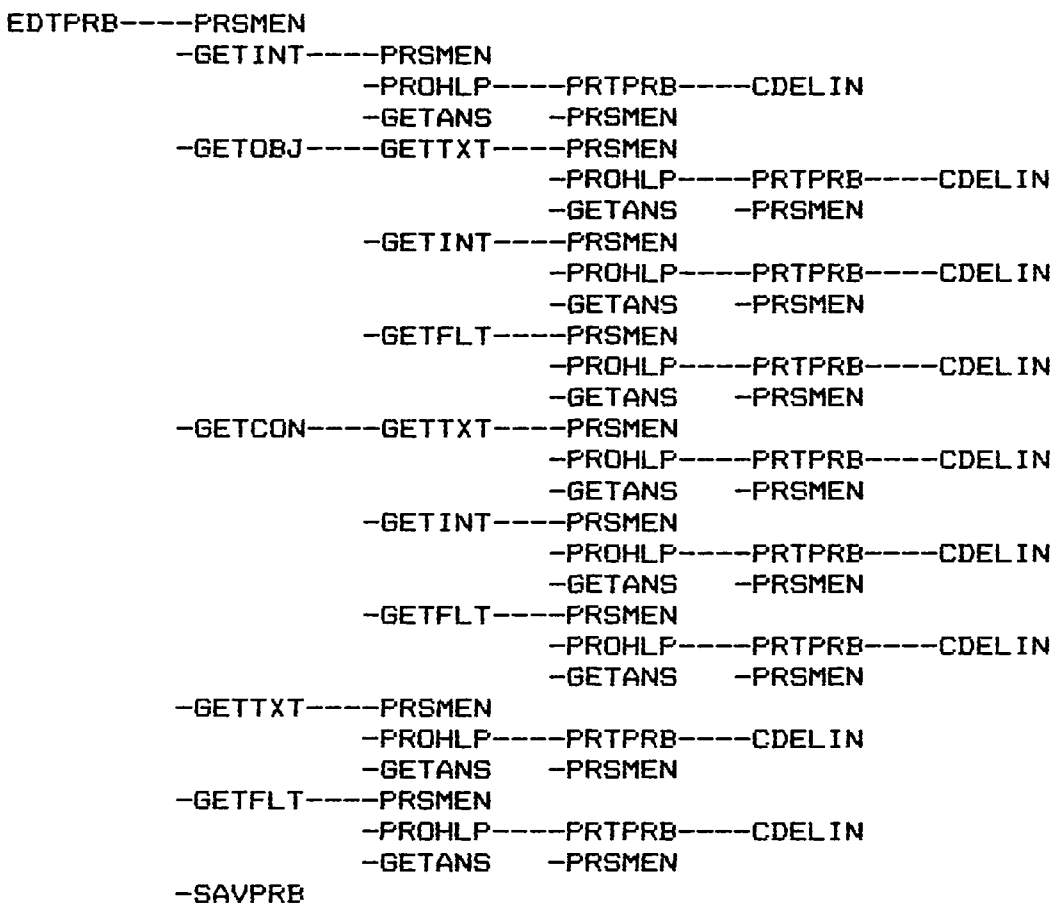


Figure 16: Problem Modification Subroutine Tree

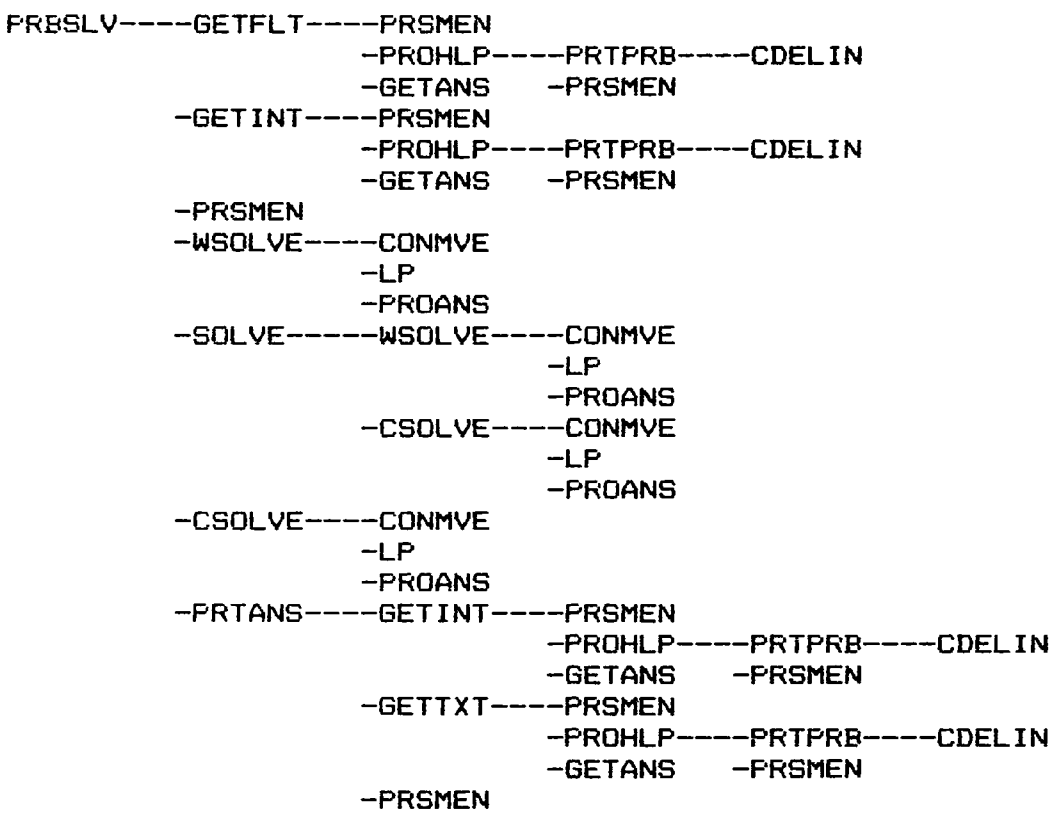


Figure 17: Problem Solution Subroutine Tree

Subroutine Cross Reference

	CDELIN	CONMVE	CSOLVE	DSKPRB	EDTPRB	GETANS	GETCON	GETFLT	GETINT	GETOBJ	GETPRB	GETTXT	LP	NEWPRB	PCADA	PRBSLV	PROANS	PROHLP	PRSMEN	PRTANS	PRTPRB	SAVPRB	SOLVE	WSOLVE
CDELIN																								
CONMVE																								
CSOLVE		X											X											
DSKPRB					X										X									
EDTPRB					X	X	X	X		X														
GETANS																								
GETCON							X	X		X														
GETFLT				X										X										
GETINT				X										X										
GETOBJ							X	X		X														
GETPRB																								
GETTXT				X										X										
LP																								
NEWPRB					X	X		X	X	X	X													
PCADA							X	X		X		X	X											
PRBSLV			X				X	X								X	X			X	X			
PROANS																								
PROHLP																								
PRSMEN																								
PRTANS																								
PRTPRB	X																							
SAVPRB																								
SOLVE																								
WSOLVE																								

Figure 18: Subroutine Cross Reference Map

To determine which subroutines are called by a particular module, read across. To determine which subroutines call some module, read down.

V Program Module Descriptions

This chapter presents the details concerning each PCADA processing module. A total of 24 modules are described, including the 23 subroutines and one main program (PCADA). Modules are presented in alphabetical order. Each section contains a processing description, a descriptive list of the module's key local variables, and the program's compliation listing.

The actual module listings can be used for most program analysis as they are all heavily commented and are written in a structured top-down format. Each listing includes detailed descriptions of all calling parameters.

CDELIN -- Code Line

Processing Description. Subroutine CDELIN is a utility function that is called repeatedly when the user has requested a display (either printed or to the video display screen) of his current problem. The purpose of CDELIN is to convert floating point constraint or objective coefficients into the appropriate character format for display. After conversion, CDELIN inserts the converted data at a position in the output line buffer as specified by the calling program.

All displays that involve coefficients also display the corresponding variable name (i.e., x1). For this, CDELIN also converts the variable number and stores it with the corresponding value. A 12-character field is used by CDELIN for these conversions. The first nine characters contain the character representation of the floating point value, which include two decimal places, and the sign. The last three character positions consist of an "x" followed by the variable number. For example, the following is a typical CDELIN conversion: " +523.07x2 ".

Key Local Variables. Refer to the module listing.

Module Listing. The following pages contain the FORTRAN compilation listing of CDELIN.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1             SUBROUTINE CDELIN ( FP, LINE2, IPNT, J )
2       C
3       C  Subroutine CDELIN -- Code line
4       C
5       C  This subroutine is called by PRTPRB to format a floating point number
6       C  and place it in a logical array at the specified point.  Floating
7       C  point numbers are formatted in a 12 character space as follows:
8       C  The first nine characters contain the sign, the number ( including
9       C  the decimal point and two decimal digits ).  NOTE:  Numbers pro-
10      C  cessed will not exceeed 99999.00 or be less than -99999.00.  The
11      C  last 3 characters contain a "x" followed by the variable number as
12      C  specified in the calling parameters.  All unused character positions
13      C  are set to blank.
14      C
15      C  parameter descriptions:
16      C
17      C  FP   -- Contains the floating point number that is to be formatted.
18      C  LINE2-- Is the ( up to 48 character ) array that will recieve the
19      C          formatted value.
20      C  IPNT -- Is a pointer to the starting character position in LINE2 that
21      C          is to receive the formatted data.
22      C  J    -- Is the variable number.
23      C
24            REAL FP, CON(10,10), OBJ(4,10), RHS(10)
25            REAL*8 CNAME(10), ONAME(4), FNAME
26            INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
27            LOGICAL LINE2(48), ITEM(12)
28            COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
29           1                NOBJ, NVAR, FNAME
30      C
31      C  Convert the floating point number to ASCII
32      C
33            ENCODE ( ITEM, 900 ) FP
34      C
35      C  Check for a positive value and if positive, search for the first
36      C    non blank character position for insertion of the "+" sign.
37      C
38            IF ( FP .LT. 0.0 ) GO TO 15
39            DO 10 K = 1, 9
40            IF ( ITEM(K) .EQ. 32 ) GO TO 10
41            II = K - 1
42            ITEM(II) = 43
43            GO TO 15
44           10 CONTINUE
45           15 CONTINUE
46      C
47      C  Store the "x" and the variable number
48      C
49            ITEM(10) = 120
```

```
50            ITEM(11) = J + 48
51            ITEM(12) = 32
52            IF ( J .NE. 10 ) GO TO 18
53            ITEM(11) = 49
54            ITEM(12) = 48
55         18 CONTINUE
56      C
57      C  Move the formatted item to the output array
58      C
59            DO 20 K = 1, 12
60            LINE2(IPNT) = ITEM(K)
61            IPNT = IPNT + 1
62         20 CONTINUE
63      C
64      C  Increment the pointer to the output array.
65      C
66        900 FORMAT ( F9.2 )
67            RETURN
68            END

Program Unit Length=00F5 (245) Bytes
Data Area Length=0022 (34) Bytes

Subroutines Referenced:

$I1                   $AT                   $OE
$ND                   $L1

Variables:

FP      0001*         LINE2   0003*         IPNT    0005*
J       0007*         CON     /PROB/+0000   OBJ     /PROB/+0212
RHS     /PROB/+01EA   CNAME   /PROB/+0190   ONAME   /PROB/+02B2
FNAME   /PROB/+02D9   CTYPE   /PROB/+01E0   OTYPE   /PROB/+02D2
NCON    /PROB/+02D6   NOBJ    /PROB/+02D7   NVAR    /PROB/+02D8
ITEM    0009*         T:000002        0015*   K       0016*
II      0018*         T:000000        001A*

COMMON  Length

/PROB/02E1 (737)

Labels:

900L    001C*         15L     0074'         10L     0067'
18L     00B2'         20L     00E3'
```

CONMVE -- Move Constraint

Processing Description. The linear programming subroutine (LP) used by PCADA requires that the problem constraints be ordered according to type. The greater than or equal constraints must be first, the equality constraints second, and the less than or equal constraints last. Since PCADA does not require users to enter the constraints in any particular order and since constraints are maintained internally in the order entered rather than by type, the constraints must be reordered prior to calling LP. The reordering process takes place as the constraints are moved from the problem array CON to the LP constraint array D for solution processing. CONMVE is called to scan CTYPE, which is parallel to CON, for constraints of the type specified by the calling program and then move them to D begining at the entry specified by the calling program. Prior to each solution, CONMVE is called three times. First to scan CTYPE for greater than or equal constraints, second for equality constraints and third for less than or equal constraints. CONMVE also counts how many constraints were moved and returns that information back to the calling program.

Key Local Variables. Refer to the module listing.

Module Listing. The following pages contain the FORTRAN compilation listing for CONMVE.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE CONMVE ( ITYPE, ICNT, II )
2       C
3       C  Subroutine CONMVE -- Move constraint
4       C
5       C  This subroutine moves constraints from the PROB common block to the
6       C  SOLN common block.  It is called repeatedly by WSOLVE and CSOLVE in
7       C  preparation for problem solution.  Each time it is called, it scans
8       C  the constraints for those constraints whose type (CTYPE) matches
9       C  ITYPE.  It moves such constraints into D sequentially beginning at
10      C  the entry indicated by II.  The subroutine also keeps track of how
11      C  many constraints have matched the specified type and have been moved.
12      C
13      C  Parameter descriptions ----
14      C
15      C  ITYPE -- Identifies the type of constraint that is to be moved during
16      C           this call.  Types are:   1 = less than or equal;
17      C                                    2 = equal;
18      C                                    3 = greater than or equal.
19      C  ICNT  -- Contains the count of how many constraints were moved from
20      C           PROB to SOLN.
21      C  II    -- Input and output as a pointer to the arrays in SOLN that
22      C           recieve the constraint data, in both cases, this item con-
23      C           tains the next available entry.
24      C
25           REAL W(4), UB(4), LB(4), DEL(4), P(30), D(10,31)
26           REAL ZF(4), OBJVAL(30,4), VARVAL(30,10)
27           REAL SOLWHT(30,4), UBWHT(4)
28           REAL CON(10,10), OBJ(4,10), RHS(10)
29           REAL*8 CNAME(10), ONAME(4), FNAME
30           INTEGER*1 SPCFLG, VARNUM(30,10), CNXREF(10)
31           INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
32           INTEGER IBV(10)
33           COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
34          1                NOBJ, NVAR, FNAME
35           COMMON / SOLN / W, UB, LB, DEL, ISTRT, P, D, IBV, SOLWHT, VARNUM,
36          1                VARVAL, OBJVAL, ZF, NNDS, CNXREF, SPCFLG, UBWHT,
37          2                ICAP
38      C
39      C  initialize ICNT to indicate that no constraints were moved.
40      C
41           ICNT = 0
42      C
43      C  enter loop to search CTYPE for constraints of the appropriate type.
44      C
45           DO 100 I = 1, NCON
46           IF ( CTYPE(I) .NE. ITYPE ) GO TO 100
47      C
48      C  for constraints of the appropriate type, enter loop to move the con-
49      C    straint coefficients from CON ( in PROB ) to D ( in SOLN ).
```

```
50      C
51            CNXREF(II) = I
52            DO 50 J = 1, NVAR
53            D(II,J) = CON(I,J)
54          50 CONTINUE
55      C
56      C  move the constraints right hand side value from RHS ( in PROB ) to D
57      C    ( in SOLN ).
58      C
59            D(II,31) = RHS(I)
60      C
61      C  increment the pointer to D ( in SOLN )
62      C
63            II = II + 1
64      C
65      C  increment the count of the number of constraints that were moved.
66      C
67            ICNT = ICNT + 1
68        100 CONTINUE
69            RETURN
70            END

Program Unit Length=010C (268) Bytes
Data Area Length=0014 (20) Bytes


Variables:

ITYPE   0001"          ICNT    0003"          II      0005"
W       /SOLN/+0000    UB      /SOLN/+0010    LB      /SOLN/+0020
DEL     /SOLN/+0030    P       /SOLN/+0042    D       /SOLN/+08BA
ZF      /SOLN/+0F42    OBJVAL  /SOLN/+0D62    VARVAL  /SOLN/+08B2
SOLWHT  /SOLN/+05A6    UBWHT   /SOLN/+0F5F    CON     /PROB/+0000
OBJ     /PROB/+0212    RHS     /PROB/+01EA    CNAME   /PROB/+0190
ONAME   /PROB/+02B2    FNAME   /PROB/+02D9    SPCFLG  /SOLN/+0F5E
VARNUM  /SOLN/+0786    CNXREF  /SOLN/+0F54    CTYPE   /PROB/+01E0
OTYPE   /PROB/+02D2    NCON    /PROB/+02D6    NOBJ    /PROB/+02D7
NVAR    /PROB/+02D8    IBV     /SOLN/+0592    ISTRT   /SOLN/+0040
NNDS    /SOLN/+0F52    ICAP    /SOLN/+0F6F    I       0007"
T:000002        0009"  T:000000        000A"  J       000C"
T:010000        000E"  T:020000        0010"  T:030000        0012"

COMMON  Length

/PROB/02E1 (737)
/SOLN/0F71 (3953)

Labels:

100L    00F7'          50L     00A3'
```

CSOLVE -- Solve Using Constraint Technique

Processing Description. This module is responsible for calling the linear programming subroutine (LP) to obtain a problem solution using the constraint technique. Prior to invoking CSOLVE, SOLVE has determined from the user the desired constraint limits and increments. CSOLVE then sets up the problem in the format required by LP, and then calls LP for solution. Included in CSOLVE's set up process is to include all but the first objectives as equality constraints. For these, the constraint limit is determined by SOLVE prior to the call to CSOLVE. Following the call to the linear programming package, PROANS is called to examine the LP solution to determine if it should be included in the non-dominated solution set. CSOLVE is called once by SOLVE for each iteration. The number of iterations is determined by the constraint limits and increments as specified by the user.

Key Local Variables.

IMULT -- Flag that is used to convert objective coefficients when the objective is to be minimized. That is, since LP only maximizes, for objectives to be minimized, the coefficients need to be multiplied by minus 1. For minimize problems, IMULT will assume the value of -1, for maximize problems, it will be set to 1.

Module Listing. The following pages contain the FORTRAN compilation listing of CSOLVE.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE CSOLVE
2       C
3       C  Subroutine CSOLVE -- Solve problem using the constraint technique
4       C
5       C  This subroutine processes problems for solution using the constraint
6       C  technique.  It is called by SOLVE to iteratively call LP with the
7       C  different problem formulations for finding the non-dominated sol-
8       C  utions.  Prior to calling this subroutine, SOLVE has determined
9       C  the constraint limits and the number of steps to go between them.
10      C  After each call to LP, this program saves the results and compares
11      C  them to obtain the set of nondominated solutions.
12      C
13      C  This subroutine has no calling parameters.
14      C
15           REAL W(4), UB(4), LB(4), DEL(4), CON(10,10), OBJ(4,10), RHS(10)
16           REAL P(30), D(10,31)
17           REAL SOLWHT(30,4), UBWHT(4)
18           REAL ZF(4), OBJVAL(30,4), VARVAL(30,10)
19           INTEGER IBV(10)
20           REAL*8 CNAME(10), ONAME(4), FNAME
21           INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
22           INTEGER*1 SPCFLG, VARNUM(30,10), CNXREF(10)
23           COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
24          1                NOBJ, NVAR, FNAME
25           COMMON / SOLN / W, UB, LB, DEL, ISTRT, P, D, IBV, SOLWHT, VARNUM,
26          1                VARVAL, OBJVAL, ZF, NNDS, CNXREF, SPCFLG, UBWHT,
27          2                ICAP
28      C
29      C  set IMULT to -1 if the objective is minimize, otherwise, set it to
30      C    1.  This is necessary because LP processes only MAXIMIZE problems.
31      C
32           IMULT = 1
33           IF ( OTYPE(1) .EQ. 2 ) IMULT = -1
34      C
35      C  enter loop to move the objective coefficients from OBJ ( in PROB ) to
36      C    P ( in SOLN ).
37      C
38           DO 100 I = 1, NVAR
39           P(I) = OBJ(1,I) * IMULT
40         100 CONTINUE
41      C
42      C  initialize the pointer to the constraint array, D ( in SOLN ).
43      C
44           II = 1
45      C
46      C  call CONMVE to move the greater than or equal constraints from CON
47      C    ( in PROB ) to D ( in SOLN ).
48      C
49           CALL CONMVE ( 3, IA, II )
```

```
50            IF ( NOBJ .EQ. 1 ) GO TO 210
51      C
52      C  enter loop to move the remaining objectives into the constraint array
53      C    as "equal" constraints.
54      C
55            DO 200 I = 2, NOBJ
56            D(II,31) = W(I)
57            DO 150 J = 1, NVAR
58            D(II,J) = OBJ(I,J)
59          150 CONTINUE
60            II = II + 1
61          200 CONTINUE
62          210 CONTINUE
63      C
64      C  call CONMVE to move the "equal" constraints from CON ( in PROB ) to
65      C    D ( in SOLN ).
66      C
67            CALL CONMVE ( 2, NEQ, II )
68      C
69      C  set NEQ equal to the number of equal constraints...remember, this
70      C    includes the real equal constraints AND all but one of the
71      C    objectives ( that have been processed into equal constraints ).
72      C
73            NEQ = NEQ + NOBJ - 1
74      C
75      C  call CONMVE to move the "less than or equal" constraints from CON
76      C    ( in PROB ) to D ( in SOLN ).
77      C
78            CALL CONMVE ( 1, IX, II )
79      C
80      C  set the number of constraints
81      C
82            IW = NCON + NOBJ - 1
83      C
84      C  set the number of variables.
85      C
86            IY = NVAR + 1
87      C
88      C  set the number of matrix columns, including one for the right hand
89      C    side.
90      C
91            IZ = IY + NEQ + IX + ( 2 * IA )
92      C
93      C  call LP to get the solution to the current formulation.
94      C
95            CALL LP ( IW, IZ, IY, IA, NEQ )
96            CALL PROANS ( IZ, IW )
97            RETURN
98            END
Program Unit Length=018D (397) Bytes
Data Area Length=0022 (34) Bytes
```

Subroutines Referenced:

$L1	$MA	$TI
CONMVE	$M9	LP
PROANS		

Variables:

W	/SOLN/+0000	UB	/SOLN/+0010	LB	/SOLN/+0020
DEL	/SOLN/+0030	CON	/PROB/+0000	OBJ	/PROB/+0212
RHS	/PROB/+01EA	P	/SOLN/+0042	D	/SOLN/+00BA
SOLWHT	/SOLN/+05A6	UBWHT	/SOLN/+0F5F	ZF	/SOLN/+0F42
OBJVAL	/SOLN/+0D62	VARVAL	/SOLN/+08B2	IBV	/SOLN/+0592
CNAME	/PROB/+0190	ONAME	/PROB/+02B2	FNAME	/PROB/+02D9
CTYPE	/PROB/+01E0	QTYPE	/PROB/+02D2	NCON	/PROB/+02D6
NOBJ	/PROB/+02D7	NVAR	/PROB/+02D8	SPCFLG	/SOLN/+0F5E
VARNUM	/SOLN/+0786	CNXREF	/SOLN/+0F54	ISTRT	/SOLN/+0040
NNDS	/SOLN/+0F52	ICAP	/SOLN/+0F6F	IMULT	0001"
T:000002	0003"	I	0004"	T:000000	0006"
II	0008"	IA	000A"	J	000C"
T:010000	000E"	T:020000	0010"	NEQ	0012"
IX	0014"	IW	0016"	IY	0018"
IZ	001A"				

COMMON Length

/PROB/02E1 (737)
/SOLN/0F71 (3953)

Labels:

100L	0047'	210L	0113'	200L	00FF'
150L	00E4'				

DSKPRB -- Retrieve Problem from Disk

Processing Description. All problems defined by users with PCADA are automatically saved to disk when they are defined. In order to retrieve them, the PCADA system includes the necessary software functions to retrieve and store problem disk files. These functions are isolated to two program modules. SAVPRB is used to store problem files on disk and GETPRB is used to retrieve problem files from disk. Prior to invoking GETPRB to retrieve a problem file, DSKPRB is called to ask the user to identify the desired file by name. After obtaining the user's response, GETPRB is called. If GETPRB encounters some kind of input/output error that prevents the loading of the requested file, DSKPRB informs the user and asks him if he wants to try again. If the user responds yes, the process is repeated, otherwise control reverts to the calling program.

Key Local Variables. Refer to module listing.

Module Listing. The following pages contain the FORTRAN compilation listing for DSKPRB.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE DSKPRB
2       C
3       C  Subroutine DSKPRB -- Retrieve problem from disk
4       C
5       C  This subroutine is designed to obtain a previously defined problem
6       C  from a disk file and load it into the common block / PROB /.  Disk
7       C  files containing problems must be installed in disk drive number 2.
8       C  Also, problem file names are xxxxxxxx.PRB where xxxxxxxx is defined
9       C  by the user.  DSKPRB will return control only after a problem has
10      C  been load or after the user has expressed his desire to not load a
11      C  problem.
12      C
13      C  there are no input/output parameters for this subroutine
14      C
15           LOGICAL TXTVAL(32), NAME(8)
16           REAL*8  CNAME(10), ONAME(4), FNAME
17           INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
18           REAL CON(10,10), OBJ(4,10), RHS(10)
19           COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
20          1                NOBJ, NVAR, FNAME
21           EQUIVALENCE ( TXTVAL(1), NAME(1) )
22          5 CONTINUE
23      C
24      C  Ask the user for a filename via GETTXT, note: filenames can be from
25      C   two to eight characters in length, the user need not specify the
26      C   "PRB".
27      C
28           CALL GETTXT ( 36, 0, 8, 2, TXTVAL, 0 )
29      C
30      C  Call GETPRB to attempt to retrieve the requested file from disk.
31      C    Upon return, ISTAT indicates if the retrieval was successful.
32      C
33           CALL GETPRB ( NAME, ISTAT )
34      C
35      C  At this point, an ISTAT value of 1 indicates some disk problem has
36      C    occurred.
37      C
38           IF ( ISTAT .NE. 1 ) GO TO 9999
39      C
40      C  Inform the user that a problem has occured, give him the options,
41      C    and ask him if he wants to try again.
42      C
43           CALL GETINT ( 42, 0, 2, 1, INTVAL )
44      C
45      C  A user response of one (INTVAL) indicates that the user understands
46      C    the problem and wants to try again, check for one and branch back
47      C    to try again.
48      C
49           IF ( INTVAL .EQ. 1 ) GO TO 5
```

```
50       9999 CONTINUE
51            RETURN
52            END

Program Unit Length=0062 (98) Bytes
Data Area Length=0034 (52) Bytes

Subroutines Referenced:

GETTXT                GETPRB                GETINT

Variables:

TXTVAL  0001"         NAME    0001"         CNAME   /PROB/+0190
ONAME   /PROB/+02B2   FNAME   /PROB/+02D9   CTYPE   /PROB/+01E0
OTYPE   /PROB/+02D2   NCON    /PROB/+02D6   NOBJ    /PROB/+02D7
NVAR    /PROB/+02D8   CON     /PROB/+0000   OBJ     /PROB/+0212
RHS     /PROB/+01EA   ISTAT   0029"         T:000002        002B"
INTVAL  002C"

COMMON  Length

/PROB/02E1 (737)

Labels:

5L      0000'         9999L   0049'
```

EDTPRB -- Edit Problem

Processing Description. EDTPRB handles all problem edit functions that are offered by the PCADA system. When the user indicates that he wants to modify the current problem, EDTPRB is called by PCADA. The first task of EDTPRB is to determine which editing function is desired. The following options are available:

-- Add/delete objective function
-- Add/delete constraint
-- Add/delete variable
-- Edit objective coefficients
-- Edit constraint coefficients
-- Edit constraint right hand side

After the user indicates which edit function is desired, EDTPRB prompts the user for all information necessary to make the modification. Following each edit operation, the user is given the opportunity to perform do other edit operations. When the user indicates that no more modifications are desired, EDTPRB calls SAVPRB to save the modified problem to disk. The modified problem is saved using the original file name thus, the original file is lost.

Key Local Variables.

F914 and F917 -- These arrays contain format statements that require subtle modifications depending on the problem. The technique of using data defined format statements saves considerable space over the alternative of requiring several slighty different format statements.

Module Listing. The following pages contain the FORTRAN compilation listing for EDTPRB.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1             SUBROUTINE EDTPRB
2       C
3       C  Subroutine EDTPRB -- Edit Problem
4       C
5       C  This subroutine handles all EDIT functions applicable to current
6       C  problems.  It is invoked by PCADA whenever the user specifies that
7       C  he wants to EDIT his problem.  The user can select from here, one or
8       C  all editing functions and when done requests exit.  Nine editing
9       C  functions are handled.  They are:
10      C
11      C     1.  Add objective function
12      C     2.  Delete objective function
13      C     3.  Add constraint
14      C     4.  Delete constraint
15      C     5.  Add variable
16      C     6.  Delete variable
17      C     7.  EDIT objective coefficients
18      C     8.  EDIT constraint coefficients
19      C     9.  EDIT constraint right hand side
20      C
21      C  this subroutine has no calling parameters....
22      C
23            INTEGER F914(17)
24            INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
25            REAL CON(10,10), OBJ(4,10), RHS(10), F917(14)
26            REAL*8 CNAME(10), ONAME(4), FNAME, NAME
27            LOGICAL TXTVAL(32), F914X(34), F917X(56)
28            COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
29           1                NOBJ, NVAR, FNAME
30            EQUIVALENCE ( TXTVAL(1), NAME ), ( F914(1), F914X(1) ),
31           1            ( F917(1), F917X(1) )
32            DATA F914 / '(1', 'H ', ',1', '9H', ' V', 'AR', 'IA', 'BL', 'E ',
33           1            'AD', 'DE', 'D ', '= ', 'x,', 'I-', ',/', ') ' /
34            DATA F917 / '(1H ', ',I2,', '28H.', ' Coe', 'ffic', 'ient',
35           1            ' for', ' var', 'iabl', 'e x,', 'I-,2', 'H= ,',
36           2            'F11.', '5)  '  /
37          10 CONTINUE
38      C
39      C  Clear the screen and write introductory message
40      C
41            WRITE ( 5, 900 )
42          12 CONTINUE
43            CALL PRSMEN ( 95, 0 )
44      C
45      C  ask the user which of the 9 edit functions (or exit) he wants to do
46      C
47            CALL GETINT ( 101, 0, 9, 0, INPRO )
48            WRITE ( 5, 900 )
49            GO TO ( 100, 200, 300, 400, 500, 600, 700, 800, 800 ),INPRO
```

```
50          GO TO 899
51        100 CONTINUE
52      C
53      C  ----process add objective function----
54      C
55      C  If there are already four objectives, then none can be added.
56      C
57          IF ( NOBJ .NE. 4 ) GO TO 110
58          WRITE ( 5, 901 )
59          GO TO 12
60        110 CONTINUE
61          II = NOBJ + 1
62          WRITE ( 5, 902 ) II, FNAME
63      C
64      C  call to add objective function
65      C
66          CALL GETOBJ ( II )
67          GO TO 10
68        200 CONTINUE
69      C
70      C  ----process delete objective----
71      C
72      C  If there is only one objective left, you can't delete it.
73      C
74          IF ( NOBJ .NE. 1 ) GO TO 205
75          WRITE ( 5, 905 )
76          GO TO 12
77        205 CONTINUE
78      C
79      C  enter loop to list the objective functions in the current problem
80      C
81          WRITE ( 5, 903 )
82          DO 210 I = 1, NOBJ
83          WRITE ( 5, 906 ) I, ONAME(I)
84        210 CONTINUE
85          II = NOBJ
86      C
87      C  ask user which of the objective functions he wants to delete.
88      C
89          CALL GETINT ( 107, 0, II, 0, INTVAL )
90          IF ( INTVAL .EQ. 0 ) GO TO 10
91      C
92      C  if he wanted to delete the last one, no problem...otherwise, must
93      C    uppack the rest.
94      C
95          IF ( INTVAL .EQ. NOBJ ) GO TO 290
96          II = INTVAL + 1
97          DO 220 I = II, NOBJ
98          III = I - 1
99          ONAME(III) = ONAME(I)
100         OTYPE(III) = OTYPE(I)
```

```
101           DO 215 J = 1, NVAR
102           OBJ(III,J) = OBJ(I,J)
103         215 CONTINUE
104         220 CONTINUE
105         290 CONTINUE
106       C
107       C  for successful deletions, decrement the number of objective functions
108       C
109           NOBJ = NOBJ - 1
110           GO TO 10
111         300 CONTINUE
112       C
113       C  ----process add constraint----
114       C
115       C  if there are already 10 constraints, no more can be added.
116       C
117           IF ( NCON .NE. 10 ) GO TO 310
118           WRITE ( 5, 907 )
119           GO TO 12
120         310 CONTINUE
121           II = NCON + 1
122           WRITE ( 5, 908 ) II, FNAME
123       C
124       C  add the constraint
125       C
126           CALL GETCON ( II )
127           GO TO 10
128         400 CONTINUE
129       C
130       C  ----process delete constraint----
131       C
132       C  if there is only one constraint left, can't delete it.
133       C
134           IF ( NCON .NE. 1 ) GO TO 405
135           WRITE ( 5, 911 )
136           GO TO 12
137         405 CONTINUE
138       C
139       C  enter loop to list the available constraints for the user
140       C
141           WRITE ( 5, 904 )
142           DO 410 I = 1, NCON
143           WRITE ( 5, 906 ) I, CNAME(I)
144         410 CONTINUE
145           II = NCON
146       C
147       C  ask the user which constraint he wants to delete.
148       C
149           CALL GETINT ( 112, 0, II, 0, INTVAL )
150           IF ( INTVAL .EQ. 0 ) GO TO 10
151       C
```

```
C  if he wants to delete the last one ok.  Otherwise, the others must
C    be uppacked.
C
      IF ( INTVAL .EQ. NCON ) GO TO 490
      II = INTVAL + 1
      DO 420 I = II, NCON
      III = I - 1
      CNAME(III) = CNAME(I)
      CTYPE(III) = CTYPE(I)
      RHS(III) = RHS(I)
      DO 415 J = 1, NVAR
      CON(III,J) = CON(I,J)
  415 CONTINUE
  420 CONTINUE
  490 CONTINUE
C
C  for successful deletions, decrement the number of constraints
C
      NCON = NCON - 1
      GO TO 10
  500 CONTINUE
C
C  ----process add variables----
C
C  If there are already 10 variables, then no more can be added.
C
      IF ( NVAR .NE. 10 ) GO TO 505
      WRITE ( 5, 912 )
      GO TO 12
  505 CONTINUE
C
C  increment the variable counter and initialize all applicable
C    data to 0.0.
C
      NVAR = NVAR + 1
      DO 510 I = 1, NVAR
      CON(I,NVAR) = 0.0
  510 CONTINUE
      DO 515 I = 1, NOBJ
      OBJ(I,NVAR) = 0.0
  515 CONTINUE
C
C  Inform user of the number of the variavle that was added, let him
C    know that the coefficients have been initialized to zero.
C
      F914X(30) = X'31'
      IF ( NVAR .EQ. 10 ) F914X(30) = X'32'
      WRITE ( 5, F914 ) NVAR
      CALL PRSMEN ( 117, 0 )
      CALL GETTXT ( 122, 0, 32, 0, TXTVAL, 1 )
      GO TO 10
```

```
  600 CONTINUE
C
C  ----process delete variable----
C
C  If there is only one variable left, it can not be deleted.
C
      IF ( NVAR .NE. 1 ) GO TO 605
      WRITE ( 5, 913 )
      GO TO 12
  605 CONTINUE
      II = NVAR
C
C  Ask user which variable he wants to delete
C
      CALL GETINT ( 125, 0, II, 0, INTVAL )
      IF ( INTVAL .EQ. 0 ) GO TO 10
C
C  If he has requested deletion of the last variable, fine.  Otherwise
C    we must uppack the remaining variables.
C
      IF ( INTVAL .EQ. NVAR ) GO TO 690
      II = INTVAL + 1
      DO 620 J = II, NVAR
      JJ = J - 1
      DO 610 I = 1, NCON
      CON(I,JJ) = CON(I,J)
  610 CONTINUE
      DO 615 I = 1, NOBJ
      OBJ(I,JJ) = OBJ(I,J)
  615 CONTINUE
  620 CONTINUE
  690 CONTINUE
C
C  Decrement the number of variables.
C
      NVAR = NVAR - 1
      GO TO 10
  700 CONTINUE
C
C  ----process EDIT objective coefficients----
C
      WRITE ( 5, 919 )
C
C  Give a list of the available objective functions, then ask the user
C    which one he wants to EDIT.
C
      DO 710 I = 1, NOBJ
      WRITE ( 5, 906 ) I, ONAME(I)
  710 CONTINUE
      II = NOBJ
      CALL GETINT ( 130, 0, II, 0, INTVAL )
```

```
254           IF ( INTVAL .EQ. 0 ) GO TO 10
255         715 CONTINUE
256           WRITE ( 5, 919 )
257           WRITE ( 5, 916 ) INTVAL, ONAME(INTVAL)
258       C
259       C  Enter loop to generate a menu containing the current value for
260       C    each coefficient in the selected objective.
261       C
262           DO 730 I = 1, NVAR
263           F917X(42) = X'31'
264           IF ( I .EQ. 10 ) F917X(42) = X'32'
265           WRITE ( 5, F917 ) I, I, OBJ(INTVAL,I)
266         730 CONTINUE
267           II = NVAR
268       C
269       C  Ask the user which coefficient to EDIT.
270       C
271           CALL GETINT ( 135, 0, II, 0, INT2 )
272           IF ( INT2 .EQ. 0 ) GO TO 700
273       C
274       C  Ask the user for the new coefficient value.
275       C
276           CALL GETFLT ( 140, 0, 99999.0, -99999.0, OBJ(INTVAL,INT2) )
277           GO TO 715
278         800 CONTINUE
279       C
280       C  ----process EDIT constraint----
281       C      (NOTE: both coefficient and RHS edits begin here.
282       C
283           WRITE ( 5, 920 )
284       C
285       C  Enter loop to list the constraints in the current problem.
286       C
287           DO 810 I = 1, NCON
288           WRITE ( 5, 906 ) I, CNAME(I)
289         810 CONTINUE
290           II = 143
291           IF ( INPRO .EQ. 9 ) II = 148
292           III = NCON
293       C
294       C  Ask the user which co    C
296           CALL GETINT ( II, 0, III, 0, INTVAL )
297           IF ( INTVAL .EQ. 0 ) GO TO 10
298         81599           WRITE ( 5, 920 )
300       C
301       C  At this point RHS processing branches.
302       C
303           IF ( INPRO .EQ. 9 ) GO TO 850
304           WRITE ( 5, 921 ) INTVAL, CNAME(INTVAL)
305       C
306       C  Enter loop to generate a menu showing each constraint coefficient.
```

```
C
      DO 830 I = 1, NVAR
      F917X(42) = X'31'
      IF ( I .EQ. 10 ) F917X(42) = X'32'
      WRITE ( 5, F917 ) I, I, CON(INTVAL,I)
  830 CONTINUE
      II = NVAR
C
C  Ask the user which coefficient to change.
C
      CALL GETINT ( 135, 0, II, 0, INT2 )
      IF ( INT2 .EQ. 0 ) GO TO 800
C
C  Ask user for lue
C
      CALL GETFLT ( 140, 0, 99999.0, -99999.0, CON(INTVAL,INT2) )
      GO TO 815
  850 CONTINUE
C
C  Show user the current right hand side value (RHS)
C
      WRITE( 5, 923TVAL, RHS(INTVAL)

C  Ask user for a new right hand side value.
C
      CALL GETFLT ( 154, 0, 99999.0, 0.0, RHS(INTVAL) )
      GO TO 800
  899 CONTINUE
C
C  Save the modified problem to disk.
C
      CALL SAVPRB ( ISTAT )
  900 FORMAT ( 1H1 )
  901 FORMATLREADY HAS THE MAXIMUM OF 4 ',
     1            'OBJECTIVE FUNCTIONS ****',/)
  902 FORMAT ( 1H ,'ADDING OBJECTIVE',I2,' TO PROBLEM ',A8,/)
  903 FORMAT ( 1H , 'DELETE OBJECTIVE', / )
  904 FORMAT ( 1H , 'DELETE CONSTRAINT', / )
  905 FORMAT ( 1H ,'**** YOUR PROBLEM CURRENTLY HAS THE MINIMUM ',
     1            'ALLOWABLE 1 OBJECTIVE ****',/)
  906 FORMAT ( 1H ,'     ',I2,'.  ',A8)
  907 FORMAT ( 1H ,'**** YOUR PROBLEM ALREADY HAS THE MAXIMUM OF 10',
     1            ' CONSTRAINTS ****',/)
  908 FORMAT ( 1H ,'ADDING CONSTRAINT',I2,' TO PROBLEM ',A8,/)
  911 FORMAT ( 1H ,'**** YOUR PROBLEM CURRENTLY HAS THE MINIMUM ',
     1            'ALLOWABLE 1 CONSTRAINT ****',/)
  912 FORMAT ( 1H ,'**** YOUR PROBLEM ALREADY HAS THE MAXIMUM OF 10',
     1            ' VARIABLES ****',/)
  913 FORMAT ( 1H ,'**** YOUR PROBLEM ALREADY HAS THE MINIMUM OF 1',
     1            ' VARIABLE ****',/)
  916 FORMAT ( 1H ,'Following are the objective coefficients for ',
```

```
358          1            'objective',I1,': ',A8,/)
359       919 FORMAT ( 1H1,'EDIT OBJECTIVE FUNCTION',/)
360       920 FORMAT ( 1H1,'EDIT CONSTRAINT',/)
361       921 FORMAT ( 48H Following are the coefficients for constraint,
362          1         I2, 2H: , A8, / )
363       923 FORMAT ( 37H The right hand side for constraint, I2, 4H is ,
364          1         F11.5, / )
365           RETURN
366           END

Program Unit Length=0A2F (2607) Bytes
Data Area Length=0480 (1152) Bytes

Subroutines Referenced:

$I3                  $I2                  $I1
$I0                  $W2                  $ND
PRSMEN               GETINT               $CG
GETOBJ               $L3                  $T3
$L1                  $T1                  GETCON
$M9                  GETTXT               GETFLT
SAVPRB

Variables:

F914    0001"        CTYPE   /PROB/+01E0   OTYPE   /PROB/+02D2
NCON    /PROB/+02D6  NOBJ    /PROB/+02D7   NVAR    /PROB/+02D8
CON     /PROB/+0000  OBJ     /PROB/+0212   RHS     /PROB/+01EA
F917    0023"        CNAME   /PROB/+0190   ONAME   /PROB/+02B2
FNAME   /PROB/+02D9  NAME    005B"         TXTVAL  005B"
F914X   0001"        F917X   0023"         INPRO   007B"
T:000002        0083"  II      0084"       I       0086"
T:000000        0088"  INTVAL  008A"       III     0092"
T:010000        0094"  J       0096"       T:020000        0098"
T:030000        009A"  JJ      00B0"       INT2    00B8"
ISTAT   00DE"

COMMON  Length

/PROB/02E1 (737)

Labels:

10L     0000'        900L    00E0"        12L     000C'
100L    0049'        200L    009D'        300L    0211'
400L    0265'        500L    03FA'        600L    04F4'
700L    062C'        800L    07BC'        899L    09DC'
110L    0067'        901L    00E5"        902L    013A"
205L    00BA'        905L    01A0"        903L    016B"
210L    00FC'        906L    01F4"        290L    0207'
220L    01F3'        215L    01DF'        310L    022F'
```

907L	020D"	908L	025B"	405L	0282'
911L	028D"	904L	0185"	410L	02C4'
490L	03F0'	420L	03DC'	415L	03C8'
505L	0418'	912L	02E2"	510L	0453'
515L	0498'	916L	0511'	913L	032E"
690L	0622'	620L	060E'	610L	05AC'
615L	05FA'	919L	03C8"	710L	066E'
715L	06AB'	916L	0378"	730L	074F'
920L	03E9"	810L	07FE'	815L	085E'
850L	098C'	921L	0402"	830L	091C'
923L	0444"				

GETANS -- Get Answer

Processing Description. GETANS is called by the input processors (GETINT, GETFLT, and GETTXT) to obtain the next input item from the user. GETANS will obtain the input item from the previously read input record if it contains unprocessed data, if not, it will read a new input record from the standard input device (logical unit #5). Before examining the input, GETANS converts all lower case inputs to upper case. At this point, GETANS scans the input record for the next input item. A description of syntax rules used by GETANS appears in the program listing.

In addition to locating the input item, GETANS classifies the item as to type (i.e., integer, floating point, or text). GETANS also examines all inputs to determine if the user has requested HELP or if he has requested a display of the current problem. HELP is indicated if the user enters "HELP" or "?", and a problem display is indicated if the user enters "@" or "@P". If one of these conditions is detected, GETANS will set the calling parameter IHELP to 1 (for HELP) or 2 (for display).

Key Local Variables.

IPNT -- This item is used as a counter of the number of characters in the input item currently being processed.

Module Listing. The following pages contain the FORTRAN compilation listing for GETANS.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE GETANS ( INDATA, ITYPE, INTVAL, FPVAL, TXTVAL, ILEN,
2           1                    II, IHELP )
3       C
4       C  Subroutine GETANS -- Get Answer
5       C
6       C  This subroutine is called to obtain a value from the input stream.
7       C  It can either be called to read a new card image, or to  process
8       C  data from the previously read card image.  GETANS maintains a
9       C  pointer to the most recently read card image.  GETANS classifies
10      C  all inputs into one of three types: INTEGER, FLOATING POINT or TEXT.
11      C  All digits is INTEGER, all digits with one decimal point is
12      C  FLOATING POINT, and everything else is TEXT.  For INTEGER and
13      C  FLOATING POINT, a leading plus (+) or minus (-) sign is allowed.
14      C  Several syntax rules apply: a) each value must be less than 32
15      C  characters in length, after 32 characters, an 'invisible' delimiter
16      C  is assumed. b) commas and/or blanks serve as delimiters, and all
17      C  other special characters are legal text data. c) the input record
18      C  is limited to 128 characters. d) leading blanks are ignored. e)
19      C  INTEGERS must be in the range -32767 to +32767, larger integer
20      C  values are converted to floating point, but are = 0 as integers.
21      C  f) FLOATING POINT values must be 15 digits or less.
22      C
23      C  Parameter descriptions ----
24      C
25      C  INDATA -- logical array to contain the input record
26      C  ITYPE  -- returned to calling program, identifying input type
27      C            1 = INTEGER
28      C            2 = FLOATING POINT
29      C            3 = TEXT
30      C            4 = illegal INTEGER (ie > 32767 )
31      C            5 = illegal FLOATING POINT (ie > 15 digits )
32      C  INTVAL -- the integer value of the input, if applicable
33      C  FPVAL  -- the floating point value of the input, if applicable
34      C  TXTVAL -- the text value of the input
35      C  ILEN   -- the length of the input (characters)
36      C  II     -- pointer to INDATA, if = 1 then GETANS will read a new
37      C            record
38      C  IHELP  -- indicates that the user has requested help
39      C
40            LOGICAL TXTVAL(32), INDATA(128), TEMP(8)
41            INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
42            REAL CON(10,10), OBJ(4,10), RHS(10)
43            REAL*8 CNAME(10), ONAME(4), FNAME
44            COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
45           1                NOBJ, NVAR, FNAME
46            EQUIVALENCE ( FNAME, TEMP(1) )
47      C
48      C  Check to see if a new input record should be read.
49      C
```

```
      10 CONTINUE
         IF ( II .NE. 1 ) GO TO 50
         WRITE ( 5, 1002 )
         READ(5,1000,ERR=50) (INDATA(I),I=1,128)
      50 CONTINUE
C
C  initialize variables
C
         FPVAL = 0.0
         IHELP = 0
         INTVAL = 0
         ILEN = 0
         ITYPE = 1
         IPNT = 1
C
C  check to see if the end of the current input record has been
C     reached, if so, return defaults as input.
C
         IF ( II .GT. 128 ) GO TO 250
C
C  enter loop to isolate the next value on the input record.
C
         DO 200 I = II, 128
C
C  Upshift lowercase inputs
C
         IF ( INDATA(I) .GE. 97 .AND. INDATA(I) .LE. 122 )
        1          INDATA(I) = INDATA(I) - 32
         III = I + 1
C
C  Look for leading blanks
C
         IF ( INDATA(I) .EQ. X'20' .AND. IPNT .EQ. 1 ) GO TO 200
C
C  Check for comma or blank (indicates next array element)
C
         IF ( INDATA(I) .EQ. X'2C' .OR. INDATA(I) .EQ. X'20' ) GO TO 210
C
C  Check for period (.) indicates floating point value
C
         IF ( INDATA(I) .EQ. X'2E' ) GO TO 180
C
C  Check for leading minus sign (-) indicating negative value
C
         IF ( INDATA(I) .EQ. X'2D' .AND. IPNT .EQ. 1 ) GO TO 198
C
C  Check for a digit (0,1,2,3,4,5,6,7,8,9)
C
         IF ( INDATA(I) .LT. X'30' .OR.
        1     INDATA(I) .GT. X'39' ) ITYPE = 3
         GO TO 198
```

```
  180 CONTINUE
C
C  increment type count.  If type was an integer, then finding a
C     decimal point promotes it to floating, if it was already
C     floating, then finding another decimal promotes it to text.
C
      IF ( ITYPE .LT. 3 ) ITYPE = ITYPE + 1
  198 CONTINUE
C
C  save the next character and increment the count
C
      TXTVAL(IPNT) = INDATA(I)
      IPNT = IPNT + 1
C
C  if current value is 33 then assume delimiter and cut off the value
C
      IF ( IPNT .EQ. 33 ) GO TO 210
  200 CONTINUE
  210 CONTINUE
C
C  prepare for number conversion, if type found was text, then
C    conversion does not apply.
C
      IF ( ITYPE .EQ. 3 ) GO TO 240
C
C  check for a value that can not be converted (ie, greater than
C    15 characters)
C
      IF ( IPNT .LE. 15 ) GO TO 215
      ITYPE = ITYPE + 3
      GO TO 250
  215 CONTINUE
C
C  store trailing blanks in the array that is about to be decoded
C    as floating point.
C
      DO 217 I = IPNT, 15
      TXTVAL(I) = X'20'
  217 CONTINUE
C
C  if value found was an integer, add a decimal point for the
C    conversion routine.
C
      IF ( ITYPE .EQ. 1 ) TXTVAL(IPNT) = X'2E'
      DECODE ( TXTVAL,1001 ) FPVAL
      TXTVAL(IPNT) = X'20'
C
C  see if floating point value exceeds the capacity of an integer
C    variable
C
      IF ( ABS(FPVAL) .GT. 32767.0 ) GO TO 220
```

```
152           INTVAL = FPVAL
153           GO TO 250
154         220 CONTINUE
155       C
156       C  set type to indicate illegal integer was found
157       C
158           IF ( ITYPE .EQ. 1 ) ITYPE = 4
159         240 CONTINUE
160       C
161       C  Check to see if the text input is a request for HELP
162       C
163           IF ( IPNT .EQ. 5 .AND.
164          1    TXTVAL(1) .EQ. X'48' .AND.
165          2    TXTVAL(2) .EQ. X'45' .AND.
166          3    TXTVAL(3) .EQ. X'4C' .AND.
167          4    TXTVAL(4) .EQ. X'50' )     IHELP = 1
168           IF ( IPNT .EQ. 2 .AND. TXTVAL(1) .EQ. X'3F' ) IHELP = 1
169           IF ( TEMP(1) .EQ. X'2F' ) GO TO 250
170           IF ( IPNT .EQ. 2 .AND. TXTVAL(1) .EQ. X'40' ) IHELP = 2
171           IF ( IPNT .EQ. 3 .AND. TXTVAL(1) .EQ. X'40' .AND.
172          1     TXTVAL(2) .EQ. X'50' ) IHELP = 3
173         250 CONTINUE
174       C
175       C  set item length, and save the current input pointer.
176       C
177           ILEN = IPNT - 1
178           II = III
179        1000 FORMAT(128A1)
180        1001 FORMAT(F15.5)
181        1002 FORMAT ( 1H , '--> ' )
182           RETURN
183           END

Program Unit Length=04B3 (1203) Bytes
Data Area Length=0040 (64) Bytes

Subroutines Referenced:

$I2                   $I1                   ABS
$AT                   $W2                   $ND
$R5                   $L1                   $TI
$OD                   $SB                   $CH

Variables:

INDATA  0001"         ITYPE   0003"         INTVAL  0005"
FPVAL   0007"         TXTVAL  0009"         ILEN    000B"
II      000D"         IHELP   000F"         TEMP    /PROB/+02D9
CTYPE   /PROB/+01E0   OTYPE   /PROB/+02D2   NCON    /PROB/+02D6
NOBJ    /PROB/+02D7   NVAR    /PROB/+02D8   CON     /PROB/+0000
OBJ     /PROB/+0212   RHS     /PROB/+01EA   CNAME   /PROB/+0190
```

```
ONAME   /PROB/+02B2     FNAME   /PROB/+02D9     T:000002        0011"
I       0018"           T:000000        001A"   IPNT    001C"
T:010002        001E"   III     001F"           T:010000        0021"
T:020002        0023"   T:030002        0024"   T:040002        0025"

COMMON  Length

/PROB/02E1 (737)

Labels:

10L     000F'           50L     006D'           1002L   0034"
1000L   0026"           250L    0489'           200L    0243'
210L    0250'           180L    01E6'           198L    020E'
240L    035B'           215L    0296'           217L    02AE'
1001L   002D"           220L    0339'
```

GETCON -- Get Constraint

Processing Description. GETCON is called by both EDTPRB and NEWPRB whenever a problem constraint is to be entered. GETCON prompts the user for all constraint information including, constraint name, constraint coefficients, constraint type (i.e., greater than, equal, or less than), and constraint right hand side value. The constraint is added to the problem as constraint number NCON + 1.

The first question GETCON asks the user is for the constraint name. If the user is done entering constraints (when first defining the problem) or no longer wishes to add a constraint (when editing the problem), he indicates this to PCADA by responding "/" when asked for the constraint name. If no constraint is added, GETCON sets its calling parameter (II) to -1. Upon entry to GETCON, II is set to the desired constraint number

Key Local Variables. Refer to the module listing.

Module Listing. The following pages contain the FORTRAN compilation listing for GETCON.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE GETCON ( II )
2       C
3       C  Subroutine GETCON -- Get Constraint
4       C
5       C  This subroutine is called to prompt the user for the information
6       C  needed for a constraint.  It is assumed that the constraint being
7       C  added is the currently last constraint, so NCON is adjusted
8       C  accordingly.  Control is returned when all constraint data has
9       C  been obtained or when the user says he doesn't want to define a
10      C  constraint.
11      C
12      C  parameter description --
13      C
14      C  II -- is input as the constraint number.  It is unchanged unless
15      C        the user decides against entering the constraint in which case
16      C        it is set to -1.
17      C
18           LOGICAL TXTVAL(32), F905X(56)
19           INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
20           REAL CON(10,10), OBJ(4,10), RHS(10), F905(14)
21           REAL*8 NAME, FNAME, CNAME(10), ONAME(4)
22           COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
23          1                NOBJ, NVAR, FNAME
24           EQUIVALENCE ( TXTVAL(1), NAME ), ( F905(1), F905X(1) )
25           DATA F905 / '(44H', ' Ent', 'er c', 'onst', 'rain', 't co',
26          1           'effi', 'cien', 't fo', 'r va', 'riab', 'le x',
27          2           ',1-,', '/)  '  /
28      C
29      C  ask for the constraint name (optional)
30      C
31           CALL GETTXT ( 79, 0, 8, 0, TXTVAL, 1 )
32      C
33      C  if the name is "/", then the user is done inputting constraints.
34      C
35           IF ( TXTVAL(1) .NE. X'2F' ) GO TO 30
36           II = -1
37           GO TO 9999
38          30 CONTINUE
39      C
40      C  save the number of constraints and the constraint name.
41      C
42           NCON = II
43           CNAME(II) = NAME
44      C
45      C  enter loop on the number of variables to define the coefficients for
46      C    this constraint.
47      C
48           DO 110 J = 1, NVAR
49           F905X(51) = X'31'
```

```
50            IF ( J .EQ. 10 ) F905X(51) = X'32'
51            WRITE ( 5, F905 ) J
52      C
53      C  get the constraint coefficient
54      C
55            CALL GETFLT ( 84, 1, 99999.0, -99999.0, CON(II,J) )
56          110 CONTINUE
57      C
58      C  call GETINT to ask the user what relation applies to this constraint.
59      C    1 --> less than or equal
60      C    2 --> equal
61      C    3 --> greater than or equal
62      C
63            CALL GETINT ( 87, 0, 3, 1, INTVAL )
64            CTYPE(II) = INTVAL
65      C
66      C  call GETFLT to get the right hand side for the constraint.  Note,
67      C    negative right hand sides are not allowed.
68      C
69            CALL GETFLT ( 160, 0, 99999.0, 0.0, RHS(II) )
70         9999 CONTINUE
71            RETURN
72            END

Program Unit Length=0149 (329) Bytes
Data Area Length=007E (126) Bytes

Subroutines Referenced:

$IO                   GETTXT                $L3
$T3                   $W2                   $ND
GETFLT                $M9                   GETINT

Variables:

II      0001*         TXTVAL  0003*         F905X   0023*
CTYPE   /PROB/+01E0   OTYPE   /PROB/+02D2   NCON    /PROB/+02D6
NOBJ    /PROB/+02D7   NVAR    /PROB/+02D8   CON     /PROB/+0000
OBJ     /PROB/+0212   RHS     /PROB/+01EA   F905    0023*
NAME    0003*         FNAME   /PROB/+02D9   CNAME   /PROB/+0190
ONAME   /PROB/+02B2   T:000002        0063*  T:000000        0064*
J       0066*         T:010000        0068*  INTVAL  0070*

COMMON  Length

/PROB/02E1 (737)

Labels:

30L     002B'         9999L   0118'         110L    00C1'
```

GETFLT -- Get Floating Point

Processing Description. GETFLT is one of the PCADA input processors. It is designed to present a question and obtain a floating point value in response. GETFLT will not return control to the calling program until it obtains a legal floating point value within the specified range. For each call to GETFLT, the user specifies the acceptable value range with the calling parameters. FHIGH indicates the highest acceptable value and FLOW indicates the lowest acceptable value. Illegal inputs (i.e., text or erroneous numerics) and out of range values will cause GETFLT to generate an appropriate error message. GETFLT will also call PROHLP when needed in response to user HELP or display problem requests. After a HELP, a display request, or an error message, GETFLT will reask the question and look for a legitimate floating point response. The process is continued until a legal floating point value is obtained.

Key Local Variables. Refer to the module listing.

Module Listing. The following pages contain the FORTRAN compilation listing for GETFLT.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE GETFLT ( MENUID, MENGEN, FHIGH, FLOW, FPVAL )
2       C
3       C  Subroutine GETFLT -- Get Floating Point
4       C
5       C  Subroutine GETFLT is called whenever a floating point input is to
6       C  be obtained from the user.  This subroutine will not return control
7       C  to the calling program until a legal floating point value within
8       C  the specified range has been obtained.  Erroneous inputs result in
9       C  the generation of an appropriate error message and the regeneration
10      C  of the menu or question.  If the user should input "?" or "help",
11      C  GETFLT will call PRSMEN to disply that question's help information.
12      C
13      C  parameter descriptions:
14      C
15      C  MENUID -- Contains the identification of the desired menu.
16      C  MENGEN -- Specifies whether or not the menu is to be displayed.  In
17      C            some cases, the calling program may have generated a
18      C            variable menu, and just desire the answer.  Setting
19      C            MENGEN to a nonzero value will suppress the generation
20      C            of the menu.
21      C  FHIGH  -- This item contains the highest floating point value that
22      C            is acceptable for this input.
23      C  FLOW   -- This item contains the lowest floating point value that is
24      C            acceptable for this input.
25      C  FPVAL  -- This item is returned to the calling program containing
26      C            the users input floating point value.
27      C
28           LOGICAL TXTVAL(32), INDATA(128)
29          10 CONTINUE
30      C
31      C  Determine if the menu should be displayed for this request.  If so,
32      C    call PRSMEN to present it.
33      C
34           IF ( MENGEN .NE. 0 ) GO TO 20
35           CALL PRSMEN ( MENUID, 0 )
36          20 CONTINUE
37      C
38      C  Set the input record pointer (II) to indicate that a new input
39      C    record should be read, and then call GETANS to get a response
40      C    from the user.
41      C
42           II = 1
43           CALL GETANS ( INDATA, ITYPE, INTVAL, FPVAL, TXTVAL, ILEN,
44          1              II, IHELP )
45      C
46      C  Determine if the user requested help, and if he did, call PRSMEN
47      C    to generate the help display for this menu.
48      C
49           IF ( IHELP .EQ. 0 ) GO TO 30
```

```
50            CALL PROHLP ( IHELP, MENUID )
51            GO TO 10
52          30 CONTINUE
53       C
54       C  Check for an illegal floating point input value (ie floating point
55       C    inputs must be less than 15 characters in length -- including the
56       C    sign and decimal point).
57       C
58            IF ( ITYPE .LE. 3 .OR. ( ITYPE .EQ. 4 .AND. ILEN .LE. 15 ) )
59           1      GO TO 50
60            WRITE ( 5, 900 ) ( TXTVAL(I), I = 1, ILEN )
61            WRITE ( 5, 901 )
62            GO TO 10
63          50 CONTINUE
64       C
65       C  Check to see if a text (ie. non-floating point) value was input.
66       C
67            IF ( ITYPE .NE. 3 ) GO TO 100
68            WRITE ( 5, 900 ) ( TXTVAL(I), I = 1, ILEN )
69            WRITE ( 5, 902 )
70            GO TO 10
71         100 CONTINUE
72       C
73       C  Check to insure that the floating point value that was input is
74       C    within the desired range.
75       C
76            IF ( FPVAL .LE. FHIGH .AND. FPVAL .GE. FLOW ) GO TO 9999
77            WRITE ( 5, 900 ) ( TXTVAL(I), I = 1, ILEN )
78            WRITE ( 5, 903 ) FLOW, FHIGH
79            GO TO 10
80          900 FORMAT ( 1H ,'**** INPUT ERROR ****    INPUT VALUE = ',
81           1         32A1 )
82          901 FORMAT ( 1H ,'LEGAL FLOATING POINT VALUES MUST BE 15 CHARACTERS',
83           1            ' OR LESS, PLEASE TRY AGAIN',/)
84          902 FORMAT ( 1H ,'A FLOATING POINT (OR INTEGER) INPUT IS REQUIRED',
85           1            ', PLEASE TRY AGAIN',/)
86          903 FORMAT ( 1H ,'A FLOATING POINT VALUE BETWEEN ',F15.5,' AND ',
87           1            F15.5,' IS REQUIRED.',/)
88         9999 CONTINUE
89            RETURN
90            END

Program Unit Length=01E3 (483) Bytes
Data Area Length=01F0 (496) Bytes

Subroutines Referenced:

$I2                 $I1                 $AT
PRSMEN              GETANS              PROHLP
$W2                 $ND                 $L1
$SB
```

Variables:

MENUID	0001"	MENGEN	0003"		FHIGH	0005"	
FLOW	0007"	FPVAL	0009"		TXTVAL	000B"	
INDATA	002B"	T:000002		00AB"	II	00AC"	
ITYPE	00AE"	INTVAL	00B0"		ILEN	00B2"	
IHELP	00B4"	T:010002		00C2"	T:020002		00C3"
I	00C4"	T:000000		00C6"			

Labels:

10L	000F'	20L	002F'		30L	0067'	
50L	00F4'	900L	00C8"		901L	00FD"	
100L	0151'	902L	0155"		9999L	01D6'	
903L	01A3"						

GETINT -- Get Integer

Processing Description. GETINT is one of the PCADA input processors. It is designed to present a question and obtain a integer value in response. GETINT will not return control to the calling program until it obtains a legal integer value within the specified range. For each call to GETINT, the user specifies the acceptable value range with the calling parameters. IHIGH indicates the highest acceptable value and ILOW indicates the lowest acceptable value. Illegal inputs (i.e., text or erroneous numerics) and out of range values will cause GETINT to generate an appropriate error message. GETINT will also call PROHLP when needed in response to user HELP or display problem requests. After a HELP, a display request, or an error message, GETINT will reask the question and look for a legitimate integer response. The process is continued until a legal integer value is obtained.

Key Local Variables. Refer to the module listing.

Module Listing. The following pages contain the FORTRAN compilation listing for GETINT.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE GETINT ( MENUID, MENGEN, IHIGH, ILOW, INTVAL )
2       C
3       C  Subroutine GETINT -- Get Integer
4       C
5       C  Subroutine GETINT is called whenever an integer input is to be
6       C  obtained from the user.  This subroutine will not return control
7       C  to the calling program until a legal integer within the specified
8       C  range has been obtained.  Erroneous inputs result in the generation
9       C  of an appropriate error message and regeneration of the menu or
10      C  question.  If the user should input "?" or "HELP", GETINT will
11      C  call PRSMEN to display that questions help information.
12      C
13      C  parameter descriptions:
14      C
15      C  MENUID -- Contains the identification of the desired menu.
16      C  MENGEN -- Specifies whether or not the menu is to be displayed.  In
17      C            some cases, the calling program may have generated a
18      C            variable menu, and just desire the answer.  Setting
19      C            MENGEN to a nonzero value will suppress the generation
20      C            of the menu.
21      C  IHIGH  -- This item contains the highest integer value that is
22      C            acceptable for this input.
23      C  ILOW   -- This item contains the lowest integer value that is
24      C            acceptable for this input.
25      C  INTVAL -- This item is returned to the calling program containing
26      C            the users input integer value.
27      C
28           LOGICAL TXTVAL(32), INDATA(128)
29          10 CONTINUE
30      C
31      C  Determine if the menu should be displayed for this request.
32      C    And, if so, call PRSMEN to present it.
33      C
34           IF ( MENGEN .NE. 0 ) GO TO 20
35           CALL PRSMEN ( MENUID, 0 )
36          20 CONTINUE
37      C
38      C  Set the input record pointer (II) to indicate that a new input
39      C    record should be read, and then call GETANS to get a response
40      C    from the user.
41      C
42           II = 1
43           CALL GETANS ( INDATA, ITYPE, INTVAL, FPVAL, TXTVAL, ILEN,
44          1              II, IHELP )
45      C
46      C  Determine if the user requested help, and if he did, call PRSMEN
47      C    to generate the help display for this menu.
48      C
49           IF ( IHELP .EQ. 0 ) GO TO 30
```

```
50            CALL PROHLP ( IHELP, MENUID )
51            GO TO 10
52          30 CONTINUE
53       C
54       C  If an integer was input, but it was out of legal range, go generate
55       C    the out of range error message.
56       C
57            IF ( ITYPE .EQ. 4 ) GO TO 150
58       C
59       C  Check to determine if a legal integer was input.
60       C
61            IF ( ITYPE .EQ. 1 ) GO TO 100
62       C
63       C  For non-integer inputs, generate the appropriate error message.
64       C
65            WRITE ( 5, 900 ) ( TXTVAL(I), I = 1, ILEN )
66            WRITE ( 5, 901 )
67            GO TO 10
68          100 CONTINUE
69       C
70       C  Check the integer value to make sure it is within the requested
71       C    range.
72       C
73            IF ( INTVAL .LE. IHIGH .AND. INTVAL .GE. ILOW ) GO TO 9999
74          150 CONTINUE
75       C
76       C  Generate an error message indicating that the response was outside
77       C    of the expected range.
78       C
79            WRITE ( 5, 900 ) ( TXTVAL(I), I = 1,ILEN )
80            WRITE ( 5, 902 ) ILOW, IHIGH
81            GO TO 10
82          900 FORMAT ( 1H ,'**** INPUT ERROR ****    INPUT VALUE = ',
83             1         32A1 )
84          901 FORMAT ( 1H ,'AN INTEGER INPUT IS REQUIRED, PLEASE TRY AGAIN',/)
85          902 FORMAT ( 1H ,'AN INTEGER INPUT BETWEEN ',I5,' AND ',I5,
86             1            ' IS REQUIRED, PLEASE TRY AGAIN'/)
87         9999 CONTINUE
88            RETURN
89            END

Program Unit Length=0189 (393) Bytes
Data Area Length=0187 (391) Bytes

Subroutines Referenced:

$I2                 $IO                 $AT
PRSMEN              GETANS              PROHLP
$W2                 $ND
```

Variables:

MENUID	0001"	MENGEN	0003"	IHIGH	0005"		
ILOW	0007"	INTVAL	0009"	TXTVAL	000B"		
INDATA	002B"	T:000002	00AB"	II	00AC"		
ITYPE	00AE"	FPVAL	00B0"	ILEN	00B4"		
IHELP	00B6"	I	00C4"	T:000000	00C6"		
T:010002	00C8"						

Labels:

10L	000F'	20L	002F'	30L	0067'		
150L	0124'	100L	00D9'	900L	00C9"		
901L	00FE"	9999L	017C'	902L	0136"		

GETOBJ -- Get Objective

Processing Description. GETOBJ is called by both EDTPRB and NEWPRB whenever a problem objective is to be entered. GETOBJ prompts the user for all objective information including, objective name, objective coefficients, and objective type (i.e., maximize or minimize). The objective is added to the problem as objective number NOBJ + 1.

The first question GETOBJ asks the user is for the objective name. If the user is done entering objectives (when first defining the problem) or no longer wishes to add an objective (when editing the problem), he indicates this to PCADA by responding "/" when asked for the objective name. If no objective is added, GETOBJ sets its calling parameter (II) to -1. Upon entry to GETOBJ, II is set to the desired objective number

Key Local Variables. Refer to the module listing.

Module Listing. The following pages contain the FORTRAN compilation listing for GETOBJ.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE GETOBJ ( II )
2       C
3       C  Subroutine GETOBJ -- Get objective
4       C
5       C  This subroutine is called to obtain an objective function from the
6       C  user.  The routine returns control only after all objective in-
7       C  formation is obtained or after the user indicates his desire not to
8       C  enter an objective function.
9       C
10      C  parameter description --
11      C
12      C  II -- this item contains the objective number at input and is chang-
13      C        ed to -1 only if the user decides not to input an objective.
14      C
15           LOGICAL TXTVAL(32), F902X(56)
16           INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
17           REAL CON(10,10), OBJ(4,10), RHS(10), F902(14)
18           REAL*8 NAME, FNAME, CNAME(10), ONAME(4)
19           COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
20          1                NOBJ, NVAR, FNAME
21           EQUIVALENCE ( TXTVAL(1), NAME ), ( F902(1), F902X(1) )
22           DATA F902 / '(43H', ' Ent', 'er o', 'bjec', 'tive', ' coe',
23          1            'ffic', 'ient', ' for', ' var', 'iabl', 'e x,',
24          2            'I-,/', ')   ' /
25      C
26      C  first ask for an optional name to assign to the objective
27      C
28           CALL GETTXT ( 74, 0, 8, 0, TXTVAL, 1 )
29      C
30      C  a name of "/" indicates that there are no more objectives.
31      C
32           IF ( TXTVAL(1) .NE. X'2F' ) GO TO 30
33           II = -1
34           GO TO 9999
35          30 CONTINUE
36      C
37      C  save the number of objectives and the objective name..
38      C
39           NOBJ = II
40           ONAME(II) = NAME
41      C
42      C  call GETINT to find out if this is a maximize (=1) or a minimize (=2)
43      C
44           CALL GETINT ( 67, 0, 2, 1, INTVAL )
45           OTYPE(II) = INTVAL
46      C
47      C  loop on the number of variables, to assign the coefficients for the
48      C    constraint.
49      C
```

```
50            DO 70 J = 1, NVAR
51            F902X(50) = X'31'
52            IF ( J .EQ. 10 ) F902X(50) = X'32'
53            WRITE ( 5, F902 ) J
54      C
55      C  call GETFLT to get the coefficient
56      C
57            CALL GETFLT ( 71, 1, 99999.0, -99999.0, OBJ(II,J) )
58          70 CONTINUE
59        9999 CONTINUE
60            RETURN
61            END

Program Unit Length=011C (284) Bytes
Data Area Length=0078 (120) Bytes

Subroutines Referenced:

$I0                     GETTXT                  GETINT
$L3                     $T3                     $W2
$ND                     GETFLT

Variables:

II      0001*           TXTVAL  0003*           F902X   0023*
CTYPE   /PROB/+01E0     OTYPE   /PROB/+02D2     NCON    /PROB/+02D6
NOBJ    /PROB/+02D7     NVAR    /PROB/+02D8     CON     /PROB/+0000
OBJ     /PROB/+0212     RHS     /PROB/+01EA     F902    0023*
NAME    0003*           FNAME   /PROB/+02D9     CNAME   /PROB/+0190
ONAME   /PROB/+02B2     T:000002        0063*   INTVAL  0064*
T:000000        0066*   J       006E*           T:010000        0070*

COMMON  Length

/PROB/02E1 (737)

Labels:

30L     002B'           9999L   00F3'           70L     00DF'
```

GETPRB -- Get Problem From Disk

Processing Description. DSKPRB is called to load a problem file from disk. The name of the desired file is passed as one of the calling parameters (NAME). Since all problem files are stored on disk with the file type "PRB", GETPRB concatenates the file type designator to names before attempting a disk access.

The PCADA System uses logical unit 8 for problem files. GETPRB closes logical unit 8 (via ENDFILE) before attempting to open the requested file. After opening the file, GETPRB attempts to read all problem data into common block PROB. If all reads are successful, then problem retrieval is successful and control reverts to the calling program. An unsuccessful read indicates that one of the following has occurred:

--disk drive door open
--problem file "NAME" does not exist
--diskette is bad
--problem diskette is not installed.

For any of these conditions, GETPRB sets the status flag (ISTAT) to 1 and returns control to the calling program (DSKPRB).

Key Local Variables.

IREC -- Used locally as the problem file record number. Incremented as the file is processed.

Module Listing. The following pages contain the FORTRAN compilation listing for GETPRB.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1             SUBROUTINE GETPRB ( NAME, ISTAT )
2       C
3       C  Subroutine GETPRB -- Get Problem from disk
4       C
5       C  This subroutine is designed to retrieve a legal problem from disk
6       C  and load it into the PROB common block.
7       C
8       C  Parameter descriptions ----
9       C
10      C  NAME  -- Contains the name of the desired problem file.
11      C  ISTAT -- Is returned to the calling program as the status of the
12      C           problem retireval attempt.  A value of zero indicates that
13      C           all went well and that the problem is loaded into the
14      C           PROB common block.  A nonzero value indicates that some
15      C           sort of I/O error has occured.
16      C
17            INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
18            LOGICAL FINAME(11), TEMP1(8), NAME(8)
19            REAL CON(10,10), OBJ(4,10), RHS(10)
20            REAL*8 CNAME(10), ONAME(4), FNAME
21            COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
22           1                NOBJ, NVAR, FNAME
23            EQUIVALENCE ( TEMP1(1), FNAME )
24      C
25      C  Transfer the name of the desired file to FINAME.
26      C
27            DO 4 I = 1, 8
28            FINAME(I) = NAME(I)
29           4 CONTINUE
30      C
31      C  Store "PRB" in the last three characters of FINAME.
32      C
33            FINAME(9) = X'50'
34            FINAME(10) = X'52'
35            FINAME(11) = X'42'
36      C
37      C  Initialize the status flag to indicate no problem occurred.
38      C
39            ISTAT = 0
40      C
41      C  Check to see if the problem that is desired is already active, if
42      C    it is than no further action by GETPRB is necessary.
43      C
44            DO 8 I = 1, 8
45            IF ( NAME(I) .NE. TEMP1(I) ) GO TO 9
46           8 CONTINUE
47            GO TO 9999
48           9 CONTINUE
49      C
```

```
C  In the event that some other problem was already associated with
C    logical unit 8, we must now un-associate it to allow looking at
C    other files, and to make sure that the previous file is saved
C    intact.
C
      ENDFILE 8
C
C  OPEN the new problem file
C
      CALL OPEN ( 8, FINAME, 2 )
C
C  Attempt to read the first record (number of constraints, number of
C    objectives, number of variables)
C
      READ ( 8, ERR=10, END=10, REC=1 ) NCON, NOBJ, NVAR
      GO TO 20
   10 CONTINUE
C
C  Set the first character of FNAME to "/" to indicate that a problem
C    has occured and that the PROB data can not be trusted, also set
C    ISTAT to a non-zero value to indicate that all is not well.
C
      TEMP1(1) = X'2F'
      ISTAT = 1
      GO TO 9999
   20 CONTINUE
C
C  Loop to read the objective function(s) data.
C
      DO 30 I = 1, NOBJ
      IREC = I + 1
      READ ( 8, ERR=10, END=10, REC=IREC ) ONAME(I), OTYPE(I),
     1          (OBJ(I,J), J = 1, NVAR )
   30 CONTINUE
C
C  Loop to read the contraint data
C
      DO 40 I = 1, NCON
      IREC = I + 1 + NOBJ
      READ ( 8, ERR=10, END=10, REC=IREC ) CNAME(I), CTYPE(I),
     1          ( CON(I,J), J = 1, NVAR ), RHS(I)
   40 CONTINUE
C
C  Store the new problems name into FNAME in order to keep track of
C    what is currently loaded.
C
      DO 50 I = 1, 8
      TEMP1(I) = NAME(I)
   50 CONTINUE
 9999 CONTINUE
      RETURN
```

```
101          END

Program Unit Length=0274 (628) Bytes
Data Area Length=0037 (55) Bytes

Subroutines Referenced:

$I3                  $I2                  $I1
$EN                  OPEN                 $R5
$ND                  $M9

Variables:

NAME    0001"        ISTAT   0003"        CTYPE   /PROB/+01E0
OTYPE   /PROB/+02D2  NCON    /PROB/+02D6  NOBJ    /PROB/+02D7
NVAR    /PROB/+02D8  FINAME  0005"        TEMP1   /PROB/+02D9
CON     /PROB/+0000  OBJ     /PROB/+0212  RHS     /PROB/+01EA
CNAME   /PROB/+0190  ONAME   /PROB/+02B2  FNAME   /PROB/+02D9
I       0010"        T:000000       0012"  T:010000       0014"
T:020000       0016"  T:000002       0018"  IREC    0023"
J       002B"        T:030000       002D"  T:040000       0035"

COMMON  Length

/PROB/02E1 (737)

Labels:

4L      0028'        8L      007F'        9L      008F'
9999L   0263'        10L     00C1'        20L     00D3'
30L     015F'        40L     0221'        50L     0256'
```

GETTXT -- Get Text

Processing Description. GETTXT is one of the PCADA input processors. It is designed to present a question and obtain a text value in response. GETTXT will not return control to the calling program until it obtains a legal text value within the specified length range. For each call to GETTXT, the user specifies the acceptable item length range with the calling parameters. ILONG indicates the longest acceptable item length and ISHORT indicates the shortest acceptable length. Illegal inputs and out of length range values will cause GETTXT to generate an appropriate error message. GETTXT will also call PROHLP when needed in response to user HELP or display problem requests. After a HELP, a display request, or an error message, GETTXT will reask the question and look for a legitimate text response. The process is continued until a legal text value is obtained.

Key Local Variables. Refer to the module listing.

Module Listing. The following pages contain the FORTRAN compilation listing for GETTXT.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE GETTXT ( MENUID, MENGEN, ILONG, ISHORT, TXTVAL, IJUNK )
2       C
3       C  Subroutine GETTXT -- Get Text
4       C
5       C  Subroutine GETTXT is called whenever a text input is to be obtained
6       C  from the user.  This subroutine will not return control to the
7       C  calling program until a legal text value whose length is within the
8       C  specified range is obtained.  Erroneous inputs result in the gener-
9       C  ation of an appropriate error message and the regeneration of the
10      C  menu or question.  If the user should input "?" or "HELP", GETTXT
11      C  will call PRSMEN to display the question's help information.
12      C
13      C  parameter descriptions:
14      C
15      C  MENUID -- Contains the identification of the desired menu.
16      C  MENGEN -- Specifies whether or not the menu is to be displayed.  In
17      C            some cases, the calling program may have generated a
18      C            variable menu, and just desire the answer.  Setting
19      C            MENGEN to a nonzero value will suppress the generation
20      C            of the menu.
21      C  ILONG  -- This item contains the length of the longest text item
22      C            that is acceptable for this input.
23      C  ISHORT -- This item contains the length of the shortest text item
24      C            that is acceptable for this input.
25      C  TXTVAL -- This item is returned to the calling program containing
26      C            the user's input text value.
27      C  IJUNK  -- This item indicates that any text string is allowed if
28      C            it is set to one at entry.
29      C
30            LOGICAL TXTVAL(32), INDATA(128)
31           10 CONTINUE
32      C
33      C  determine if the menu should be generated for this text request,
34      C    if so, call PRSMEN to display it.
35      C
36            IF ( MENGEN .NE. 0 ) GO TO 20
37            CALL PRSMEN ( MENUID, 0 )
38           20 CONTINUE
39            II = 1
40      C
41      C  Blank out TXTVAL before getting input.
42      C
43            DO 25 I = 1, ILONG
44            TXTVAL(I) = X'20'
45           25 CONTINUE
46      C
47      C  Call GETANS to get another input bead from the user.
48      C
49            CALL GETANS ( INDATA, ITYPE, INTVAL, FPVAL, TXTVAL, ILEN,
```

```
50          1            II, IHELP )
51      C
52      C  Check to determine if the user has requested help.
53      C
54            IF ( IHELP .EQ. 0 ) GO TO 30
55            CALL PROHLP ( IHELP, MENUID )
56            GO TO 10
57          30 CONTINUE
58      C
59      C  Determine if GETANS has found a text item
60      C
61            IF ( ITYPE .EQ. 3 .OR. IJUNK .NE. 0 ) GO TO 50
62          45 CONTINUE
63            WRITE ( 5, 900 ) ( TXTVAL(I), I = 1, ILEN )
64            WRITE ( 5, 901 )
65            GO TO 10
66          50 CONTINUE
67      C
68      C  Determine if the users input is within the specified length range
69      C
70            IF ( ILEN .LE. ILONG .AND. ILEN .GE. ISHORT ) GO TO 60
71            WRITE ( 5, 900 ) (TXTVAL(I), I = 1, ILEN )
72            WRITE ( 5, 902 ) ISHORT, ILONG
73            GO TO 10
74          60 CONTINUE
75      C
76      C  Check to see if the item was of length zero.
77      C
78            IF ( ILEN .EQ. 0 .OR. IJUNK .NE. 0 ) GO TO 70
79      C
80      C  Make sure that the text input was a legal text value.
81      C    First character must be a letter, and remaining characters must
82      C    be digits or letters, no special characters are allowed, and no
83      C    lowercase letters are allowed.
84      C
85            IF ( TXTVAL(1) .GT. X'5A' .OR.
86           1     TXTVAL(1) .LT. X'41' ) GO TO 45
87            DO 70 I = 1, ILEN
88            IF ( ( TXTVAL(I) .LE. X'5A' .AND.
89           1       TXTVAL(I) .GE. X'41' ) .OR.
90           2     ( TXTVAL(I) .LE. X'39' .AND.
91           3       TXTVAL(I) .GE. X'30' ) ) GO TO 70
92            GO TO 45
93          70 CONTINUE
94         900 FORMAT ( 1H ,'**** INPUT ERROR ****    INPUT VALUE = ',
95           1         32A1 )
96         901 FORMAT ( 1H ,'TEXT INPUTS MUST BEGIN WITH A LETTER AND MAY ',
97           1            'CONTAIN NO SPECIAL CHARACTERS',/)
98         902 FORMAT ( 1H ,'A TEXT INPUT BETWEEN ',I2,' AND ',I2,
99           1            ' CHARACTERS IN LENGTH IS REQUIRED.',/)
100     9999 CONTINUE
```

```
101           RETURN
102           END

Program Unit Length=0261 (609) Bytes
Data Area Length=018D (397) Bytes

Subroutines Referenced:

$I2                   $IO                   $AT
PRSMEN                GETANS                PROHLP
$W2                   $ND

Variables:

MENUID  0001"         MENGEN  0003"         ILONG   0005"
ISHORT  0007"         TXTVAL  0009"         IJUNK   000B"
INDATA  000D"         T:000002        008D" II      008E"
I       0090"         T:000000        0092" ITYPE   0094"
INTVAL  0096"         FPVAL   0098"         ILEN    009C"
IHELP   009E"         T:010002        00AC" T:020002        00AD"
T:030002        00AE"

Labels:

10L     000F'         20L     002F'         25L     004D'
30L     0093'         50L     0109'         45L     00C0'
900L    00AF"         901L    00E4"         60L     01A6'
902L    013B"         70L     0244'         9999L   0254'
```

LP -- Linear Programming

Processing Description. This subroutine is called to solve a single objective linear programming problem. It is called iteratively within the PCADA system as part of the multiobjective problem solution process.

LP utilizes the SIMPLEX method as described in "Linear Programming and Extension" by Wu and Coppins. As implemented, this subroutine is limited to problems with 10 constraints and 10 variables. The program expects constraint information to be stored by the calling program in the array D. The constraints must be entered in the following order:

1. greater thans or equal
2. equals
3. less thans or equal

In addition, the objective function must be in the form to be maximized. Thus in PCADA the calling program must invert (i.e. multiply by -1) MINIMIZE objectives.

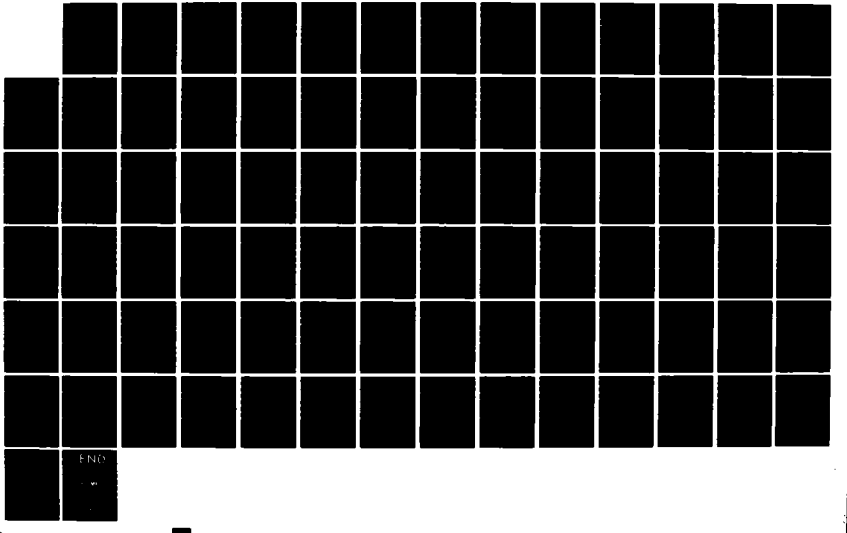
When LP is called, the RHS value for each constraint is stored in the last element of D (i.e. D(i,31)). LP moves this value to the last active column as determined by the number of constraints and variables.

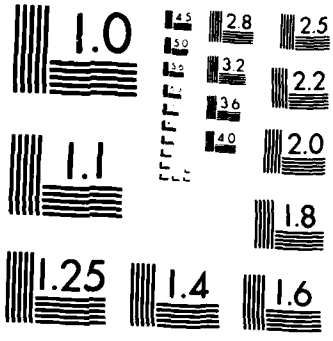
The "big M" method is used for artificial variables. LP initializes corresponding objective coefficients to -99999.0 for this purpose.

LP indicates the occurrence of a special condition through the return flag (IW). If no pivot row can be found

AD-A151 911 PERSONAL COMPUTER AIDED DECISION ANALYSIS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING G R WHITE 14 DEC 84 AFIT/GSO/OS/84D-8 3/3

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

then IW is set to -1 to indicate that the problem is unbounded. If the process terminates "normally", the basis is examined to see if it includes an artificial variable. If so, the problem is infeasible and LP indicates this by setting IW to -2. Finally, LP avoids the degeneracy problem by using the techniques suggested in Wu and Coppins.

The program does not check for the multiple optimal condition but could be adjusted to do so as noted in the comment at the program's end.

Key Local Variables.

NEG -- Defines the starting column number for the artificial variables.

NEE -- Defines the ending column number for the artificial variables.

SUM -- This variable contains the value referred to as Zj" in Wu and Coppins.

SC -- This variable contains the value referred to as "Cj - Zj" in Wu and Coppins.

SCMAX -- This variable contains the smallest "Cj - Zj" value and is used when searching for the entering variable (pivot col).

SMVAL -- This variable contains the smallest ratio (basic variable to the corresponding coefficient) that is used when searching for the leaving basic variable (pivot row).

IPIVC -- This variable identifies the pivot column.

IPIVR -- This variable identifies the pivot row.

Module Listing. The following pages contain the FORTRAN compilation listing for LP.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE LP (IW, IZ, IY, IA, NEQ )
2       C
3       C  This subprogram is invoked to solve a linear programming problem.
4       C  The SIMPLEX process is used, as described in "Linear Programming
5       C  and Extensions" by Wu & Coppins (copyright 1981).  All parameters
6       C  are input parameters except for IBV which is output and IW which
7       C  is input and output.  Parameters are used as follows:
8       C
9       C    IW is input as the number of constraints and output to indicate
10      C       the occurance of a special case:
11      C          -1 means the problem is unbounded
12      C          -2 means the solution is infeasible
13      C    IZ is the total number of matrix columns + 1
14      C       (i.e.  of variables + of equal constraints + of less than
15      C          constraints + 2 times the of greater than constraints + 1)
16      C    IY is the number of problem variables + 1
17      C    IA is the number of greater than constraints
18      C    NEQ is the number of equal constraints
19      C    P is the array of objective coefficients
20      C    D is the two dimensional array of constraints coefficients
21      C       note: D(I,IZ) contains the right hand sides
22      C    IBV is output as the array of basic variables
23      C       i.e.  IBV(3) = 5 says that X5 is in the solution and that
24      C             D(3,IZ) contains the value of X5.
25      C
26      C  This program is limited to 10 constraints, 10 variables, and the
27      C  constraints must be entered in the following order:
28      C
29      C          1. greater thans
30      C          2. equals
31      C          3. less thans
32      C
33      C  The objective function must be in a form to be maximized.
34      C  The program includes the suggestions made in Wu & Coppins
35      C  for avoiding the degeneracy problem.
36      C
37            DIMENSION P(30), D(10,31), IBV(10), SC(30)
38            INTEGER*1 I,J,M,N
39            INTEGER*1 SPCFLG, VARNUM(30,10), CNXREF(10)
40            REAL ZF(4), OBJVAL(30,4), VARVAL(30,10)
41            REAL W(4), UB(4), LB(4), DEL(4), SOLWHT(30,4), UBWHT(4)
42            COMMON / SOLN / W, UB, LB, DEL, ISTRT, P, D, IBV, SOLWHT, VARNUM,
43           1                VARVAL, OBJVAL, ZF, NNDS, CNXREF, SPCFLG, UBWHT,
44           2                ICAP
45      C
46      C  set IX equal to the number of columns in the matrix
47      C
48            IX = IZ - 1
49      C
```

```
C  initialize matrix data
C
      DO 4 I = 1, IW
      D(I,IZ) = D(I,31)
    4 CONTINUE
      DO 8 I = IY, IX
      P(I) = 0.0
      DO 6 J = 1, IW
      D(J,I) = 0.0
    6 CONTINUE
    8 CONTINUE
C
C  initialize "BIG M" coefficients for artificial variables, if any.
C
      IF ( (IA + NEQ) .EQ. 0 ) GO TO 20
      NEG = IY + IA
      NEE = NEG + IA - 1 + NEQ
      DO 10 I = NEG, NEE
      P(I) = -99999.0
   10 CONTINUE
   20 CONTINUE
C
C  initialize the identity and identify the starting basis in IBV
C
      DO 40 I = 1, IW
      J = I + IY + IA - 1
      D(I,J) = 1.0
      IBV(I) = J
   40 CONTINUE
C
C  initialize for the artificial variables
C
      IF ( IA .EQ. 0 ) GO TO 100
      DO 50 I = 1, IA
      J = I + IY - 1
      D(I,J) = -1.0
   50 CONTINUE
  100 CONTINUE
      SCMAX = 0.0
C
C  search for and select pivot column
C     note:  SUM = "Zj" and P(j) = "Cj"
C
      DO 130 I = 1, IX
      SUM = 0.0
      DO 110 J = 1, IW
      M = IBV(J)
      SUM = SUM + P(M) * D(J,I)
  110 CONTINUE
      SC(I) = P(I) - SUM
      IF ( SC(I) .LE. SCMAX ) GO TO 130
```

```
101           SCMAX = SC(I)
102           IPIVC = I
103         130 CONTINUE
104       C
105       C  select pivot row
106       C     at this point, if SCMAX = 0 then solution is complete
107       C
108           IF ( SCMAX .LE. 0.0 ) GO TO 210
109           SMVAL = 99999999.
110           IPIVR = 0
111           DO 160 M = 1, IW
112           IF ( D(M,IPIVC) .LE. 0.0 ) GO TO 160
113           QUONT = D(M,IZ) / D(M,IPIVC)
114           IF ( SMVAL .LE. QUONT ) GO TO 160
115           IPIVR = M
116           SMVAL = QUONT
117         160 CONTINUE
118       C
119       C  perform the pivot
120       C     first, adjust the pivot row, then do the pivot
121       C
122           IF ( IPIVR .EQ. 0 ) GO TO 205
123           IBV(IPIVR) = IPIVC
124           DIV = D(IPIVR,IPIVC)
125           DO 170 N = 1, IZ
126           D(IPIVR,N) = D(IPIVR,N) / DIV
127         170 CONTINUE
128           DO 200 M = 1, IW
129           IF ( M .EQ. IPIVR ) GO TO 200
130           CM = -D(M,IPIVC)
131           DO 190 N = 1, IZ
132           D(M,N) = D(M,N) + ( D(IPIVR,N) * CM )
133         190 CONTINUE
134         200 CONTINUE
135       C
136       C  now go back and do another pivot
137       C
138           GO TO 100
139         205 CONTINUE
140       C
141       C  set IW to indicate that the problem is unbounded
142       C
143           IW = -1
144           GO TO 230
145         210 CONTINUE
146       C
147       C  for normal finish, must now look to see if the basis contains an
148       C     artificial variable, if so, the problem is infeasible.
149       C
150           IF ( (IA+NEQ) .EQ. 0 ) GO TO 230
151           DO 220 I = 1, IW
```

```
152           IF ( IBV(I) .LT. NEG .OR. IBV(I) .GT. NEE ) GO TO 220
153           SUM = ABS ( D(I,IZ) )
154           IF ( SUM .LT. 0.001 ) GO TO 220
155           IW = -2
156           GO TO 230
157         220 CONTINUE
158       C
159       C  Note:  At this point, one could check for multiple optimal
160       C         by seeing if Zj - Cj = 0 for any j (non-basic
161       C         variables only).
162       C
163         230 CONTINUE
164           RETURN
165           END
```

```
Program Unit Length=0637 (1591) Bytes
Data Area Length=0083 (179) Bytes

Subroutines Referenced:
ABS                   $AT                   $M9
$L1                   $T1                   $MB
$AB                   $SB                   $DB
$NB

Variables:
IW      0001"         IZ      0003"         IY      0005"
IA      0007"         NEG     0009"         P       /SOLN/+0042
D       /SOLN/+00BA   IBV     /SOLN/+0592   SC      000B"
I       0083"         J       0084"         M       0085"
N       0086"         SPCFLG  /SOLN/+0F5E   VARNUM  /SOLN/+0786
CNXREF  /SOLN/+0F54   ZF      /SOLN/+0F42   OBJVAL  /SOLN/+0D62
VARVAL  /SOLN/+08B2   W       /SOLN/+0000   UB      /SOLN/+0010
LB      /SOLN/+0020   DEL     /SOLN/+0030   SOLWHT  /SOLN/+05A6
UBWHT   /SOLN/+0F5F   ISTRT   /SOLN/+0040   NNDS    /SOLN/+0F52
ICAP    /SOLN/+0F6F   IX      0087"         T:000000        0089"
T:010000        008B" T:000002        008D" NEG     008E"
NEE     0090"         T:020000        0092" SCMAX   0094"
SUM     0098"         IPIVC   009C"        SMVAL   009E"
IPIVR   00A2"         QUONT   00A4"        DIV     00A8"
CM      00AC"         T:030000        00B0" T:010002        00B2"

COMMON  Length
/SOLN/0F71 (3953)

Labels:
4L      0055'         8L      00C5'         6L      00BA'
20L     0146'         10L     013B'         40L     01B8'
100L    0231'         50L     0226'         130L    0311'
110L    02AC'         210L    0552'         160L    03EC'
205L    0545'         170L    0479'         200L    0537'
190L    052C'         230L    061E'         220L    0613'
```

NEWPRB -- Define New Problem

Processing Description. This routine is called when the user indicates that he wants to enter a new problem for PCADA processing. First, the user is asked for an eight character name for the problem. If the user responds "STOP", it indicates that he has changed his mind about entering a new problem, and control is returned to the calling program. For all other names, NEWPRB calls GETPRB to determine if the name is already in use. If it is, the user is given the option of using it anyway (and losing the original file) or of specifying a new name.

After an acceptable name has been obtained, NEWPRB will ask the user how many variables the problem is to contain. Next, the user is asked for the objectives and constraints.

When all problem data has been entered, NEWPRB calls SAVPRB to write the new problem file to disk.

Key Local Variables. Refer to the module listing

Module Listing. The following pages contain the FORTRAN compilation listing for NEWPRB.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE NEWPRB
2       C
3       C  Subroutine NEWPRB -- Define new problem
4       C
5       C  This subroutine is designed to allow the user to define a new
6       C  multiobjective problem to be solved by PCADA.  When called, this
7       C  subroutine will prompt the user for all of the required data and
8       C  will then save the problem to disk for later reference.
9       C
10      C  there are no input/output parameters for this subroutine
11      C
12           LOGICAL TXTVAL(32)
13           INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
14           REAL CON(10,10), OBJ(4,10), RHS(10)
15           REAL*8 NAME, FNAME, CNAME(10), ONAME(4)
16           COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
17          1                NOBJ, NVAR, FNAME
18           EQUIVALENCE ( TXTVAL(1), NAME )
19      C
20      C  clear the screen
21      C
22           WRITE ( 5, 900 )
23      C
24      C  display an opening blurb to the user describing what a problem is
25      C    and that he is now going to define one.
26      C
27           CALL PRSMEN ( 48, 0 )
28          10 CONTINUE
29      C
30      C  call GETTXT to find out what the user wants to call this problem
31      C    for saving on disc.  names can be from 2 to 8 characters in length
32      C    no special characters are allowed.
33      C
34           CALL GETTXT ( 54, 0, 8, 2, TXTVAL, 0 )
35      C
36      C  see if the user responded to the question with "STOP", indicating
37      C    he really does not want to define a new problem now.
38      C
39           IF ( TXTVAL(1) .EQ. X'53' .AND. TXTVAL(2) .EQ. X'54' .AND.
40          1     TXTVAL(3) .EQ. X'4F' .AND. TXTVAL(4) .EQ. X'50' .AND.
41          2     TXTVAL(5) .EQ. X'20' ) GO TO 9999
42      C
43      C  call GETPRB for the purpose of determining if there already is a
44      C    problem file name that is the same as what the user wants to call
45      C    this one.
46      C
47           CALL GETPRB ( NAME, ISTAT )
48      C
49      C  A status value of zero indicates that a problem was correctly loaded!
```

```
C    ie. a file already exists with the name that the user specified.
C
      IF ( ISTAT .NE. 0 ) GO TO 40
C
C  call GETINT to tell the user that the file already exists and ask him
C    if this was intentional.  If so, allow him to redefine using that
C    file, if not, ask for a new file name.
C
      CALL GETINT ( 58, 0, 2, 1, INTVAL )
      IF ( INTVAL .EQ. 2 ) GO TO 10
   40 CONTINUE
C
C  assign the user specified name to FNAME
C
      FNAME = NAME
C
C  call GETINT to ask the user how many variables his problem is to
C    contain
C
      CALL GETINT ( 63, 0, 10, 1, INTVAL )
      NVAR = INTVAL
C
C  enter loop to get the objectives...note maximum of 4.
C
      NOBJ = 0
      DO 80 I = 1, 4
      WRITE ( 5, 901 ) I
      II = 1
      CALL GETOBJ ( II )
      IF ( II .EQ. -1 ) GO TO 85
   80 CONTINUE
   85 CONTINUE
      IF ( NOBJ .NE. 0 ) GO TO 88
      WRITE ( 5, 902 )
      GO TO 10
   88 CONTINUE
      NCON = 0
C
C  loop to obtain the constraints for this problem.  note the maximum
C    of 10.
C
      DO 120 I = 1, 10
      WRITE ( 5, 904 ) I
      II = I
      CALL GETCON ( II )
      IF ( II .EQ. -1 ) GO TO 125
  120 CONTINUE
  125 CONTINUE
      IF ( NCON .NE. 0 ) GO TO 130
      WRITE ( 5, 902 )
      GO TO 10
```

```
101       130 CONTINUE
102     C
103     C  call SAVPRB to save the newly defined problem to disk.
104     C
105           CALL SAVPRB ( ISTAT )
106       900 FORMAT ( 1H1 )
107       901 FORMAT ( 1H1, 'OBJECTIVE FUNCTION ',I1,/)
108       902 FORMAT ( 1H ,'**** INVALID PROBLEM-MUST HAVE AT LEAST ONE ',
109          1            'OBJECTIVE AND CONSTRAINT ****',/)
110       904 FORMAT ( 1H1, 'CONSTRAINT ',I2,/)
111      9999 CONTINUE
112           RETURN
113           END
```

Program Unit Length=01CA (458) Bytes
Data Area Length=00DA (218) Bytes

Subroutines Referenced:

$I0	$W2	$ND
PRSMEN	GETTXT	GETPRB
GETINT	$L3	$T3
GETOBJ	GETCON	SAVPRB

Variables:

TXTVAL	0001"	CTYPE	/PROB/+01E0	OTYPE	/PROB/+02D2
NCON	/PROB/+02D6	NOBJ	/PROB/+02D7	NVAR	/PROB/+02D8
C8N	/PROB/+0000	OBJ	/PROB/+0212	RHS	/PROB/+01EA
NAME	0001"	FNAME	/PROB/+02D9	CNAME	/PROB/+0190
ONAME	/PROB/+02B2	T:000002	0029"	T:010002	002A"
T:020002	002B"	T:030002	002C"	T:040002	002D"
ISTAT	002E"	INTVAL	0030"	I	003E"
II	0040"				

COMMON Length

/PROB/02E1 (737)

Labels:

900L	0042"	10L	0015'	9999L	01A1'
40L	00A9'	80L	0108'	901L	0047"
85L	0115'	88L	0131'	902L	0069"
120L	0172'	904L	00C0"	125L	017F'
130L	019B'				

PCADA -- Main Program

Processing Description. PCADA is the system's main program. Its purpose is to initialize the system, open the necessary files, determine what the user wants to do, and then call the appropriate function.

First, PCADA opens the menu/help file and assigns logical unit 7 to it. Next, PCADA presents the user with some general instructions and then asks him to specify whether he wants to load a file from disk or define a new one. At this point, these are the only options since a problem must be identified before it can be solved or modified. Once a problem has been identified, the user is presented with system level menu 2. This menu gives the user the following options:

- load a new problem from disk
- define a new problem
- edit the problem
- solve the problem

Based on the user's response, PCADA calls the appropriate function. When the requested process is completed, control again reverts to PCADA where the user is again faced with system level menu 2 for his next request. Processing continues in this fashion until the user indicates he wishes to exit.

Key Local Variables. Refer to the Module Listing.

Module Listing. The following pages contain the FORTRAN compilation listing for PCADA.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            PROGRAM PCADA
2       C*************************************************************************
3       C*                                                                       *
4       C*  PROGRAM PCADA -- PERSONAL COMPUTER AIDED DECISION ANALYSIS           *
5       C*                                                                       *
6       C*  This program was written as part of thesis effort in partial        *
7       C*  fulfillment of the requirements for the degree of Master of          *
8       C*  Science in Space Operations from the Air Force Institute of          *
9       C*  Technology, December 1984.                                           *
10      C*                                                                       *
11      C*  Author:  Greg R. White                                               *
12      C*  Academic Advisor:  Lt. Col. Mark Mekaru                              *
13      C*  Thesis Reader: Maj. Ken Feldman                                      *
14      C*                                                                       *
15      C*  The program is designed to solve complex multiobjective decision     *
16      C*  analysis problems.  The objectives of the research were to demo-     *
17      C*  strate that such a program could be written for a personal com-      *
18      C*  puter in a user friendly, easy to use fashion.  This program         *
19      C*  was originally written in CP/M based FORTRAN 80.  Although           *
20      C*  first hosted on an Apple II Plus computer, the program does not      *
21      C*  take advantage of any special Apple features, and can thus be        *
22      C*  rehosted to any comparable Z80 based computer system.                *
23      C*                                                                       *
24      C*************************************************************************
25      C
26            INTEGER ACTMEN, IBV(10)
27            LOGICAL TXTVAL(32), MENVAL(768), TEMP(8)
28            INTEGER*1 SPCFLG, VARNUM(30,10), CNXREF(10)
29            INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR, MENSIZ, MENLIN,
30           1          HLPSIZ, HLPLIN, MLNSZ(12), HLNSZ(12)
31            REAL ZF(4), OBJVAL(30,4), VARVAL(30,10), SOLWHT(30,4), UBWHT(4)
32            REAL W(4), UB(4), LB(4), DEL(4), P(30), D(10,31)
33            REAL CON(10,10),OBJ(4,10),RHS(10)
34            REAL*8 CNAME(10),ONAME(4),FNAME
35      C
36      C  The following common block ( PROB ) is used throughout this program
37      C  as a storage area for data to describe the problem currently under
38      C  consideration by the user.
39      C
40      C  CON   -- This 10 by 10 array is used to contain the constraint
41      C           coefficients.  The dimensions correspond to the fact that
42      C           there are a maximum of 10 constraints with a maximum of 10
43      C           variables each.
44      C
45      C  CNAME -- This 10 element array is used to contain the constraint
46      C           name.  This optional name is input be the user when the
47      C           problem is created.
48      C
49      C  CTYPE -- This 10 element array is used to contain the constraint
```

```
C          relationship as follows:
C                   1 = less than or equal constraint
C                   2 = equal constraint
C                   3 = greater than or equal constraint
C
C  RHS   -- This 10 element array is used to contain the constraint
C          right hand side values.
C
C  OBJ   -- This 4 by 10 element array contains the objective co-
C          efficients.  There can be up to 4 objective functions.
C
C  ONAME -- This 4 element array is used to contain the objective
C          name.  This optional name is input by the user when the
C          problem is created.
C
C  OTYPE -- This 4 element array is used to contain the objective type
C          as follows:
C                   1 = MAXIMIZE
C                   2 = MINIMIZE
C
C  NCON  -- This item contains the number of constraints in the current
C          problem.
C
C  NOBJ  -- This item contains the number of objective functions in the
C          current problem.
C
C  NVAR  -- This item contains the umber of variables in the current
C          problem.
C
C  FNAME -- This item contains the name of the current problem.  Prob-
C          lems are stored on disk using this name with a file type of
C          "PRB".
C
      COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
     1                NOBJ, NVAR, FNAME
C
C  The following common block ( MENDAT ) contains information relating
C  to the currently "active" menu.
C
C  MENSIZ -- This item contains the number of records that the current
C           menu occupies on disk.  NOTE: disk records are 128 char-
C           acters long.
C
C  MENLIN -- This item contains the number of lines that the current
C           menu occupies on the screen.
C
C  HLPSIZ -- This item contains the number of records that the HELP
C           message associated with the current menu occupies on disk.
C
C  HLPLIN -- This item contains the number of lines that the HELP mes-
C           sage associated with the current menu occupies on the
```

```
101     C           screen.
102     C
103     C  MLNSZ  -- This 12 element array contains the size, in characters of
104     C            each of the MENLIN lines of the current menu.  NOTE: the
105     C            dimension of this array limits the number of lines in each
106     C            menu to 12.
107     C
108     C  HLNSZ  -- This 12 element array contains the size, in characters of
109     C            each of the HLPLIN lines of the HELP message associated
110     C            with the current menu.
111     C
112     C  MENVAL -- This 768 element array contains the actual characters that
113     C            make up the current menu and its help message.  The 768
114     C            elements corresponds to 6-128 character records.  The first
115     C            MENSIZ records ( i.e. 128 * MENSIZ characters ) contains
116     C            the menu data and the next HLPSIZ records contains the HELP
117     C            data.
118     C
119     C  ACTMEN -- This item contains the identity of the current menu.  Menus
120     C            are identified by there starting diskfile record number.
121     C
122           COMMON /MENDAT / MENSIZ, MENLIN, HLPSIZ, HLPLIN, MLNSZ, HLNSZ,
123          1                 MENVAL, ACTMEN
124     C
125     C  The following common block contains information that pertains to
126     C  the solution to the current problem.
127     C
128     C  W      -- This 4 element array contains the weights or constraint
129     C             limits (depending on solution technique).
130     C
131     C  UB     -- This 4 element array contains the upper limit for the
132     C             corresponding objective weights or constraint limit.
133     C
134     C  LB     -- This 4 element array contains the lower limit for the
135     C             corresponding objective weights or constraint limit.
136     C
137     C  DEL    -- This 4 element array contians the increment ( or delta )
138     C             that is to be used in going from t    ower limit (LB) to the
139     C             upper limit.
140     C
141     C  ISTRT -- This item identifies the solution type.
142     C                       1 = weighting technique
143     C                       2 = constraint technique
144     C
145     C  P      -- This array contains the aggregate objective coefficients
146     C             as used by the LP routine.
147     C
148     C  D      -- This array contains the constraint coefficients as used by
149     C             the LP routine.
150     C
151     C  IBV    -- This array is return from the LP routine containing the
```

```
C          identity of the basic variables in the solution.
C
C  SOLWHT-- This array serves as a storage area for the objective
C          function weights that correspond to each solution.
C
C  VARNUM-- This array contains the variable numbers for each basic
C          variable in the corresponding solution.
C
C  VARVAL-- This array is parallel to the array VARNUM and contains the
C          corresponding basic variable values for each solution.
C
C  OBJVAL-- This array contains the objective function values for the
C          corresponding solution.
C
C  ZF     -- This array contains the objective values for the current
C          solution.  Once the current solution is determined to be a
C          NON-DOMINATED solution, then ZF is moved to OBJVAL.
C
C  NNDS   -- This item contains the number of NON-DOMINATED solutions
C          that were found.
C
C  CNXREF-- This array contains pointers to the constraint arrays in
C          / CON / that are used to relate them to the constraint data
C          that is passed to the LP routine.
C
C  SPCFLG-- This item is used to signal the occurance of some special
C          condition for the current problem.  A value of 0 is default.
C          -1 means an infeasible solution has been detected.  -2 means
C          that an unbounded solution has been detected.
C
C  UBWHT -- This array is used only if an unbounded solution is detected
C          then, it contains the objective weights that were in use
C          when the solution was found.
C
C  ICAP   -- This item assumes a non-zero value whenever more than 30
C          NON-DOMINATED solutions to a problem are detected.  The
C          program is only capable of saving 30 solutions.
C
      COMMON / SOLN / W, UB, LB, DEL, ISTRT, P, D, IBV, SOLWHT, VARNUM,
     1                VARVAL, OBJVAL, ZF, NNDS, CNXREF, SPCFLG, UBWHT,
     2                ICAP
      EQUIVALENCE ( FNAME, TEMP(1) )
      DATA ACTMEN, TEMP(1) / -1 , X'2F' /
C
C  Open the menu/help file...NOTE: it must be in drive 2!
C
      CALL OPEN ( 7, 'MENDAT  DAT', 1 )
C
C  Clear the screen and then present some introductory information
C    then, wait for the user to hit a return to continue.
```

```
C
      WRITE ( 5, 900 )
      CALL PRSMEN ( 2, 0 )
      CALL PRSMEN ( 9, 0 )
      CALL GETTXT ( 15, 0, 32, 0, TXTVAL, 1 )
   20 CONTINUE
C
C  clear the screen
C
      WRITE ( 5, 900 )
C
C  determine if there is an active problem file.  NOTE: a slash (/) in
C    character position 1 is the indication that no problem is active.
C
      IF ( TEMP(1) .NE. X'2F' ) GO TO 30
C
C  in this case, no problem is active.  before the user can solve a
C    problem he must activate one.  Ask him how he wishes to do this.
C           1 = load from disk
C           2 = define a new problem
C           0 = exit program
C
      CALL GETINT ( 24, 0, 2, 0, INTVAL )
      GO TO ( 100, 200 ), INTVAL
      GO TO 9000
   30 CONTINUE
C
C  in this case, a problem exists, ask the user what he wants to do.
C           1 = load a different problem from disk
C           2 = define a new problem
C           3 = EDIT the currently active problem
C           4 = solve the currently active problem
C           0 = EXIT the program
C
      CALL GETINT ( 30, 0, 4, 0, INTVAL )
      GO TO ( 100, 200, 300, 500 ), INTVAL
      GO TO 9000
  100 CONTINUE
C
C  --- Process load a problem from disk ---
C
      CALL DSKPRB
      GO TO 20
  200 CONTINUE
C
C  --- Process define a new problem ---
C
      CALL NEWPRB
      GO TO 20
  300 CONTINUE
C
```

```
253     C  --- Process EDIT current problem ---
254     C
255           CALL EDTPRB
256           GO TO 20
257       500 CONTINUE
258     C
259     C  --- Process solve the current problem ---
260     C
261           CALL PRBSLV
262           GO TO 20
263       900 FORMAT(1H1)
264      9000 CONTINUE
265           WRITE ( 5, 900 )
266           CALL PRSMEN ( 18, 0 )
267      9999 CONTINUE
268           STOP
269           END

Program Unit Length=00F5 (245) Bytes
Data Area Length=0048 (72) Bytes

Subroutines Referenced:

$INIT                   OPEN                    $W2
$ND                     PRSMEN                  GETTXT
GETINT                  $CG                     DSKPRB
NEWPRB                  EDTPRB                  PRBSLV
$ST

Variables:

ACTMEN  /MENDAT/+031C   IBV     /SOLN/+0592     TXTVAL  0001*
MENVAL  /MENDAT/+001C   TEMP    /PROB/+02D9     SPCFLG  /SOLN/+0F5E
VARNUM  /SOLN/+0786     CNXREF  /SOLN/+0F54     CTYPE   /PROB/+01E0
OTYPE   /PROB/+02D2     NCON    /PROB/+02D6     NOBJ    /PROB/+02D7
NVAR    /PROB/+02D8     MENSIZ  /MENDAT/+0000   MENLIN  /MENDAT/+0001
HLPSIZ  /MENDAT/+0002   HLPLIN  /MENDAT/+0003   MLNSZ   /MENDAT/+0004
HLNSZ   /MENDAT/+0010   ZF      /SOLN/+0F42     OBJVAL  /SOLN/+0D62
VARVAL  /SOLN/+08B2     SOLWHT  /SOLN/+05A6     UBWHT   /SOLN/+0F5F
W       /SOLN/+0000     UB      /SOLN/+0010     LB      /SOLN/+0020
DEL     /SOLN/+0030     P       /SOLN/+0042     D       /SOLN/+00BA
CON     /PROB/+0000     OBJ     /PROB/+0212     RHS     /PROB/+01EA
CNAME   /PROB/+0190     ONAME   /PROB/+02B2     FNAME   /PROB/+02D9
ISTRT   /SOLN/+0040     NNDS    /SOLN/+0F52     ICAP    /SOLN/+0F6F
H:0001  0021*           T:000002        0034*   INTVAL  0035*

COMMON  Length

/PROB/02E1 (737)
/MENDAT/031E (798)
/SOLN/0F71 (3953)
```

Labels:

$$L	0006'	900L	0043'	20L	003C'
30L	0071'	100L	008F'	200L	0095'
9000L	00A7'	300L	009B'	500L	00A1'
9999L	00BC'				

PRBSLV -- Problem Solve

Processing Description. PRBSLV is the first level of the problem solving function. It is called when the user indicates he wants to solve a problem.

The first activity of PRBSLV is to determine which processing option (i.e., constraint or weighting) is desired. If the user requests the weighting technique, PRBSLV offers two options; using specific weights or a range of weights. For the specific weight option, the user is prompted for a weight for each objective. PRBSLV does not require these weights to sum to one. For the range of weights option, the user selects a weight increment from a pre-determined set. This increment will be used for determining objective weights as the problem is solved.

If the user requests the constraint technique, PRBSLV will prompt the user for upper bound, lower bound and increment for all but the first constraint.

Once these solution parameters are determined, PRBSLV calls SOLVE to continue solution processing.

Key Local Variables

ITER -- This variable is set to the number of iterations that the solution will require as defined by the user. For the weighting technique, ITER is a function of weight increment and the number of objectives. For the constraint technique, ITER is a function of the constraint increment and the number of objectives.

Module Listing. The following pages contain the FORTRAN compilation listing for PRBSLV.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) Bv Microsoft -- Bytes: 25131
Created: 27-Jun-81
1             SUBROUTINE PRBSLV
2       C
3       C  Subroutine PRBSLV -- Problem solve
4       C
5       C  This subroutine is the first level of the software designed to solve
6       C  problems.  Its purpose is to determine which solution technique
7       C  ( weighting or constraint ) the user desires, and then interagate the
8       C  user for the information needed to carry out the solution.  After
9       C  this, PRBSLV calls the appropriate solution routine for further
10      C  processing.
11      C
12      C  PRBSLV uses no calling parameters.
13      C
14            REAL INC(7), P(30), D(10,31), SOLWHT(30,4), UBWHT(4)
15            REAL ZF(4), OBJVAL(30,4), VARVAL(30,10)
16            REAL W(4), UB(4), LB(4), DEL(4), CON(10,10), OBJ(4,10), RHS(10)
17            REAL*8 CNAME(10), ONAME(4), FNAME
18            INTEGER IBV(10), ITS(28)
19            INTEGER*1 SPCFLG, VARNUM(30,10), CNXREF(10)
20            INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
21            COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
22           1                NOBJ, NVAR, FNAME
23            COMMON / SOLN / W, UB, LB, DEL, ISTRT, P, D, IBV, SOLWHT, VARNUM,
24           1                VARVAL, OBJVAL, ZF, NNDS, CNXREF, SPCFLG, UBWHT,
25           2                ICAP
26            DATA INC(1), INC(2), INC(3), INC(4), INC(5), INC(6), INC(7) /
27           1      1.000,  0.500,  0.333,  0.250,  0.200,  0.100,  0.050 /
28            DATA ITS(1), ITS(2), ITS(3), ITS(4), ITS(5), ITS(6), ITS(7) /
29           1       1,      2,      3,      4,      1,      3,      6    /
30            DATA ITS(8), ITS(9), ITS(10), ITS(11), ITS(12), ITS(13), ITS(14) /
31           1       10,     1,      4,       10,      20,      1,       5     /
32            DATA ITS(15),ITS(16),ITS(17), ITS(18), ITS(19), ITS(20), ITS(21) /
33           1       15,     35,     1,       6,       21,      56,      1     /
34            DATA ITS(22),ITS(23),ITS(24), ITS(25), ITS(26), ITS(27), ITS(28) /
35           1       11,     66,     286,     1,       21,      231,     621   /
36      C
37      C  make correction for INC(3), that is, get all significant digits.
38      C
39            ICAP = 0
40            NNDS = 0
41            SPCFLG = 0
42            INC(3) = 1.0 / 3.0
43          10 CONTINUE
44      C
45      C  clear the screen
46      C
47            WRITE ( 5, 9000 )
48          15 CONTINUE
49      C
```

```
C  call GETINT to determine which solution technique the user desires
C      1 = weighting
C      2 = constraint
C      0 = EXIT
C
       CALL GETINT ( 186, 0, 2, 0, ISTRT )
       GO TO ( 20, 1000 ), ISTRT
       GO TO 9999
    20 CONTINUE
C
C  ---this section of code processes the weighting technique.
C
C  clear the screen
C
       WRITE ( 5, 9000 )
C
C  present general message about the weighting technique, then ask the
C    user which of the two available options he wants.
C       1 = weight with specific weights
C       2 = weight with a range of weights on all objectives
C       0 = EXIT weight technique
C
       CALL PRSMEN ( 172, 0 )
       CALL GETINT ( 183, 0, 2, 0, INTVAL )
       GO TO ( 50, 500 ), INTVAL
       GO TO 10
C
C  enter loop to ask user for the weight on each objective.  NOTE: the
C    weights need not sum to one!
C
    50 CONTINUE
       DO 100 I = 1, NOBJ
       WRITE ( 5, 9001 ) I, ONAME(I)
       CALL GETFLT ( 189, 1, 1.0, 0.0, W(I) )
   100 CONTINUE
C
C  call WSOLVE to process the weighting technique solution
C
       CALL WSOLVE
       GO TO 2000
   500 CONTINUE
C
C  call GETINT to get the weight increment that the user wants
C     1=1.0;  2=0.5;  3=0.333;  4=0.25;  5=0.2;  6=0.1;  7=0.05
C     0 = EXIT the weighting technique
C
       CALL GETINT ( 178, 0, 7, 0, INTVAL )
       IF ( INTVAL .EQ. 0 ) GO TO 20
       ITER = ( ( INTVAL - 1 ) * 4 ) + NOBJ
       ITER = ITS(ITER)
C
```

```
C  set DEL equal to the increment
C
      DO 510 I = 1, NOBJ
      DEL(I) = INC(INTVAL)
  510 CONTINUE
C
C  set upper and lower bound limits for each of the objective weights
C
      DO 520 I = 1, NOBJ
      UB(I) = 1.000
      LB(I) = 0.000
  520 CONTINUE
      GO TO 1030
 1000 CONTINUE
C
C ---this section processes the constraint technique requests
C
      IF ( NOBJ .NE. 1 ) GO TO 1010
      CALL CSOLVE
      GO TO 2000
 1010 CONTINUE
C
C  clear the screen
C
      WRITE ( 5, 9000 )
      ITER = ( NCON + ( NOBJ - 1 ) ) - 10
      IF ( ITER .LE. 0 ) GO TO 1015
      WRITE ( 5, 9004 ) ITER
      GO TO 15
 1015 CONTINUE
C
C  present a general blurb telling what the constraint technique is all
C    about
C
      CALL PRSMEN ( 194, 0 )
      WRITE ( 5, 9002 ) ONAME(1)
C
C  enter loop to get the upper bound, lower bound, and increment for
C    each of the objectives that are to be processed as constraints.
C
      ITER = 1
      IF ( NOBJ .EQ. 1 ) GO TO 1030
      DO 1020 I = 2, NOBJ
      WRITE ( 5, 9003 ) I, ONAME(I)
      CALL GETFLT ( 200, 0, 99999.0, -99999.0, UB(I) )
      CALL GETFLT ( 205, 0, UB(I), -99999.0, LB(I) )
      CALL GETINT ( 210, 0, 50, 1, INTVAL )
C
C  calculate the delta (DEL) for the constraint limit increment.
C
      DEL(I) = (UB(I)-LB(I))/INTVAL
```

```
152           ITER = ITER + ( INTVAL + 1 )
153           WRITE ( 5, 9000 )
154       1020 CONTINUE
155       1030 CONTINUE
156      C
157      C  call SOLVE to continue solution processing -- NOTE: this call is
158      C     made for both the weighting and constraint techniques.
159      C
160           CALL SOLVE ( ITER )
161       2000 CONTINUE
162           CALL PRTANS
163       9000 FORMAT ( 1H1 )
164       9001 FORMAT ( 1H ,'Please specify the weight for objective', I1,
165          1         ': ', A8, / )
166       9002 FORMAT ( 1H , 'Objective1: ', A8, ' will be optimized.', / )
167       9003 FORMAT ( 1H ,'---OBJECTIVE',I1,': ',A8,'---')
168       9004 FORMAT ( 1H , '**** TOO MANY CONSTRAINTS FOR THE CONSTRAINT ',
169          1              'TECHNIQUE ****', /, ' ', 'Please redefine the ',
170          2              'problem, deleting', I2, ' of them, or use the ',
171          3              'weighting technique', / )
172       9999 CONTINUE
173           RETURN
174           END

Program Unit Length=0389 (905) Bytes
Data Area Length=01D2   66) Bytes

Subroutines Referenced:

$I3                   $I0                   $L1
$DB                   $T1                   $W2
$ND                   GETINT                $CG
PRSMEN                GETFLT                WSOLVE
CSOLVE                $SB                   $DA
$NB                   $M9                   SOLVE
PRTANS

Variables:

INC     0001"         P       /SOLN/+0042   D       /SOLN/+00BA
SOLWHT  /SOLN/+05A6   UBWHT   /SOLN/+0F5F   ZF      /SOLN/+0F42
OBJVAL  /SOLN/+0D62   VARVAL  /SOLN/+08B2   W       /SOLN/+0000
UB      /SOLN/+0010   LB      /SOLN/+0020   DEL     /SOLN/+0030
CON     /PROB/+0000   OBJ     /PROB/+0212   RHS     /PROB/+01EA
CNAME   /PROB/+0190   ONAME   /PROB/+02B2   FNAME   /PROB/+02D9
IBV     /SOLN/+0592   ITS     001D"         SPCFLG  /SOLN/+0F5E
VARNUM  /SOLN/+0786   CNXREF  /SOLN/+0F54   CTYPE   /PROB/+01E0
OTYPE   /PROB/+02D2   NCON    /PROB/+02D6   NOBJ    /PROB/+02D7
NVAR    /PROB/+02D8   ISTRT   /SOLN/+0040   NNDS    /SOLN/+0F52
ICAP    /SOLN/+0F6F   INTVAL  005B"         I       0063"
T:000000        0065" T:000002        0073" ITER    0074"
```

```
T:010000        0076'   T:020000        0078"

COMMON  Length

/PROB/02E1 (737)
/SOLN/0F71 (3953)

Labels:

10L     0022'           9000L   009C"           15L     002E'
20L     0048'           1000L   01A8'           9999L   0338'
50L     0077'           500L    00E5'           100L    00C8'
9001L   0091"           2000L   0335'           510L    0146'
520L    0191            1030L   032F'           1010L   01BC'
1015L   0204'           9004L   0127"           9002L   00CF"
1020L   0314'           9003L   0100"
```

PROANS -- Process Answer

Processing Description. This routine is called following each linear programming iteration. Its purpose is to examine the solution and determine if it belongs in the non-dominated solution set. If so, the current solution is added.

First, PROANS checks for the special cases. Infeasible solutions are ignored. If the solution is unbounded and it is the first unbounded solution found, PROANS sets SPCFLG to one and stores the weight values that were in effect for the solution.

At this point, PROANS computes the objective values for each of the problem objectives. The objective values are used to determine if the new solution should be included as part of the non-dominated solution set. To determine dominance, the new solution is compared with each solution that is already in the NDSS. If some non-dominated solution (NDS) has all of its objective values greater than the newest solution's objectives, then the new solution is dominated and can be rejected without examining the rest of the NDSS.

If all of the new solution's objectives exceed all of the objectives for one or more of the elements in the NDSS, then these elements are dominated and must be rejected and the new solution accepted. If the new solution has a higher value for at least one of the objectives than all of the

other solutions, then it is a NDS and it is added to the NDSS.

Key Local Variables.

ZF -- This array contains the objective values for the new solution.

IC1 -- The number of the current solution objectives that are less than the present NDS.

IC2 -- The number of the current solution objectives that are greater than the present NDS.

IC3 -- The number of the current solution objectives that are equal to or less than the present NDS.

Module Listing. The following pages contain the FORTRAN compilation listing for PROANS.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) Bv Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE PROANS ( IZ, IW )
2       C
3       C  Subroutine PROANS -- Process answer
4       C
5       C  This subroutine is called after each linear programming iteration.
6       C  Its purpose is to examine the current solution and determine if it
7       C  should be retained as a NON-DOMINATED solution.  The routine works
8       C  by comparing objective values of the current solution with all of the
9       C  NON-DOMINATED solutions already found.  If the new solution is
10      C  dominated by one or more of the ND solutions, it is rejected.  If it
11      C  dominates any ND solution than the dominated solution is thrown out.
12      C
13      C  Calling parameters ----
14      C
15      C  IZ -- The number of columns in the solution matrix -D- as generated
16      C        by LP.  NOTE that this includes a column for the right hand
17      C        side.  In fact, D(K,IZ) contains the basic variable solution
18      C        values for K = 1 to NCON (+NOBJ-1 for constraint tech).
19      C  IW -- Is the LP return code, and is used only if LP detected a
20      C        "funny" result such as infeasible ( = -1 ) or unbounded ( = -2)
21      C
22            REAL W(4), UB(4), LB(4), DEL(4), ZF(4), OBJVAL(30,4), P(30),
23           1     VARVAL(30,10), D(10,31), CON(10,10), OBJ(4,10), RHS(10)
24            REAL SOLWHT(30,4), UBWHT(4)
25            REAL*8 CNAME(10), ONAME(4), FNAME
26            INTEGER IBV(10), ISTRT, NNDS
27            INTEGER*1 SPCFLG, VARNUM(30,10), CNXREF(10), IC1, IC2, IC3,
28           1        CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR, NCONX
29            COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
30           1                NOBJ, NVAR, FNAME
31            COMMON / SOLN / W, UB, LB, DEL, ISTRT, P, D, IBV, SOLWHT, VARNUM,
32           1                VARVAL, OBJVAL, ZF, NNDS, CNXREF, SPCFLG, UBWHT,
33           2                ICAP
34      C
35      C  Compute the number of basic variables.  Recall, that this is equal to
36      C    the number of constraints.  The number of constraints is NCON, but
37      C    for the constraint technique ( ISTRT = 2 ) there are NOBJ - 1 extra
38      C    constraints.
39      C
40            NCONX = NCON + ( ISTRT - 1 ) * ( NOBJ - 1 )
41      C
42      C  Check for an infeasible solution.  If found, ignore this solution.
43      C
44            IF ( IW .EQ. (-2) ) GO TO 999
45      C
46      C  Check for an unbounded solution.  If found, check to see if it is the
47      C    first one.  If it is the first, retain the weights that were used
48      C    in getting it.  If not the first, ignore it.  The goal here is to
49      C    let the user know that at least one unbounded occurred and what
```

```
C    weights were used to get it.
C
      IF ( IW .NE. (-1) ) GO TO 50
      IF ( SPCFLG .NE. 0 ) GO TO 999
      SPCFLG = 1
      DO 40 I = 1, NOBJ
      UBWHT(I) = W(I)
   40 CONTINUE
      GO TO 999
   50 CONTINUE
C
C  Enter loop to compute the objective values for the current solution.
C    Do this for each objective, and store the values in ZF(object).
C
      DO 200 I = 1, NOBJ
      ZF(I) = 0.0
      DO 100 J = 1, NCONX
      IF ( IBV(J) .GT. NVAR ) GO TO 100
      IC1 = IBV(J)
      ZF(I) = ZF(I) + ( OBJ(I,IC1) * D(J,IZ) )
  100 CONTINUE
  200 CONTINUE
C
C  Check to see if any NON-DOMINATED solutions (NNDS) have yet been
C    found.
C
      IF ( NNDS .EQ. 0 ) GO TO 600
C
C  Enter loop to compare the current solution with each of the NON-
C    DOMINATED solutions already found.
C
      DO 500 I = 1, NNDS
  230 CONTINUE
      IC1 = 0
      IC2 = 0
      IC3 = 0
C
C  Enter loop to compute:
C    IC1 --> The number of the current solution objectives that are
C            less than the present NNDS.
C    IC2 --> The number of the current solution objectives that are
C            equal to or greater than the present NNDS.
C    IC3 --> The number of the current solution objectives that are
C            equal to or less than the present NNDS.
C
      DO 250 J = 1, NOBJ
      AX = OBJVAL(I,J) - 0.01
      AY = OBJVAL(I,J) + 0.01
      IF ( ( ZF(J) .LE. AX .AND. OTYPE(J) .NE. 2 ) .OR.
     1     ( ZF(J) .GE. AY .AND. OTYPE(J) .EQ. 2 ) ) IC1 = IC1 + 1
      IF ( ( ZF(J) .GT. AX .AND. OTYPE(J) .NE. 2 ) .OR.
```

```
101         1     ( ZF(J) .LT. AY .AND. OTYPE(J) .EQ. 2 ) ) IC2 = IC2 + 1
102          IF ( ( ZF(J) .LT. AY .AND. OTYPE(J) .NE. 2 ) .OR.
103         1     ( ZF(J) .GT. AX .AND. OTYPE(J) .EQ. 2 ) ) IC3 = IC3 + 1
104       250 CONTINUE
105     C
106     C  If all current objectives are less, than the current solution is
107     C    dominated and should be rejected.
108     C
109          IF ( IC1 .EQ. NOBJ ) GO TO 999
110     C
111     C  Check if not all of the current objectives exceed the present NNDS.
112     C    If this condition exists then so far, this is a new NNDS and we
113     C    must continue checking the others before ruling this one out.
114     C
115          IF ( IC2 .NE. NOBJ ) GO TO 500
116     C
117     c  If IC3 does not equal NOBJ and IC2 equals NOBJ then the current
118     C    solution dominates the present NDSS and the present NDSS should be
119     C    thrown out.
120     C
121          IF   IC3. NE. NOBJ ) GO TO 400
122     C
123     C  The following code is executed only for the case when all objectives
124     C    of the current solution match the present NNDS.  If the basic
125     C    variables are different then the solution must be retained other-
126     C    wise we can throw it out.
127     C
128          IC2 = 0
129     C
130     C  Enter loop to compare the x's.
131     C
132          DO 300 J = 1, NCONX
133          AX = ABS ( VARVAL(I,J) )
134          IF ( AX .GE. 0.001 ) GO TO 270
135          IC2 = IC2 + 1
136          GO TO 300
137       270 CONTINUE
138          DO 290 K = 1, NCONX
139          IF ( IBV(K) .NE. VARNUM(I,J) ) GO TO 290
140          AX = ABS ( VARVAL(I,J) - D(K,IZ) )
141          IF ( AX .LE. 0.01 ) IC2 = IC2 + 1
142          GO TO 300
143       290 CONTINUE
144       300 CONTINUE
145     C
146     C  If all the x's matched then we throw out the present NNDS and retain
147     C    the current solution.  If all the X's did not match then we keep
148     C    both.
149     C
150          IF ( IC2 .NE. NCONX ) GO TO 500
151       400 CONTINUE
```

```
C
C  NOTE: If the solution we intend to throw out is the last one, then
C    there is no need to formally uppack the arrays.
C
      IF ( I .EQ. NNDS ) GO TO 460
      IC3 = I + 1
C
C  Enter loop to delete an existing NNDS by uppacking the NNDS arrays.
C
      DO 450 J = IC3, NNDS
      IC2 = J - 1
      DO 430 K = 1, NOBJ
      OBJVAL(IC2,K) = OBJVAL(J,K)
      SOLWHT(IC2,K) = SOLWHT(J,K)
  430 CONTINUE
      DO 440 K = 1, NCONX
      VARNUM(IC2,K) = VARNUM(J,K)
      VARVAL(IC2,K) = VARVAL(J,K)
  440 CONTINUE
  450 CONTINUE
  460 CONTINUE
      NNDS = NNDS - 1
C
C  Since we have just uppacked the arrays, if we don't make the follow-
C    ing check, one NNDS will "sneak" by without being checked.  So here
C    I jump back within the loop.  NOTE:  That the loop variable I
C    doesn't increment, instead the loop limit ( NNDS ) has been de-
C    remented.
C
      IF ( I .LE. NNDS ) GO TO 230
  500 CONTINUE
C
C  At this point, we have determined that the current solution should be
C    retained as an NNDS.  If we already have the limit of 30, set the
C    capacity flag (ICAP), otherwise, add the solution.
C
      IF ( NNDS .NE. 30 ) GO TO 600
      ICAP = 1
      GO TO 999
  600 CONTINUE
      NNDS = NNDS + 1
      DO 700 I = 1, NOBJ
      OBJVAL(NNDS,I) = ZF(I)
      SOLWHT(NNDS,I) = W(I)
  700 CONTINUE
      DO 800 I = 1, NCONX
      VARVAL(NNDS,I) = D(I,IZ)
      VARNUM(NNDS,I) = IBV(I)
  800 CONTINUE
  999 CONTINUE
      RETURN
```

```
203          END

Program Unit Length=0776 (1910) Bytes
Data Area Length=002D (45) Bytes

Subroutines Referenced:

ABS                  $M9                  $L1
$T1                  $MB                  $AB
$MA                  $NB                  $SB

Variables:

IZ      0001*        IW      0003*        W       /SOLN/+0000
UB      /SOLN/+0010  LB      /SOLN/+0020  DEL     /SOLN/+0030
ZF      /SOLN/+0F42  OBJVAL  /SOLN/+0D62  P       /SOLN/+0042
VARVAL  /SOLN/+08B2  D       /SOLN/+00BA  CON     /PROB/+0000
OBJ     /PROB/+0212  RHS     /PROB/+01EA  SOLWHT  /SOLN/+05A6
UBWHT   /SOLN/+0F5F  CNAME   /PROB/+0190  ONAME   /PROB/+02B2
FNAME   /PROB/+02D9  IBV     /SOLN/+0592  ISTRT   /SOLN/+0040
NNDS    /SOLN/+0F52  SPCFLG  /SOLN/+0F5E  VARNUM  /SOLN/+0786
CNXREF  /SOLN/+0F54  IC1     0005*        IC2     0006*
IC3     0007*        CTYPE   /PROB/+01E0  OTYPE   /PROB/+02D2
NCON    /PROB/+02D6  NOBJ    /PROB/+02D7  NVAR    /PROB/+02D8
NCONX   0008*        ICAP    /SOLN/+0F6F  T:000002        0009*
T:010002        000A*  I       000B*        T:000000        000D*
T:010000        000F*  J       0011*        T:020000        0013*
T:030000        0015*  AX      0017*        AY      001B*
T:020002        001F*  T:030002        0020*  K       0021*
T:000001        0023*  T:040000        0027*  T:050000        0029*
T:060000        002B*

COMMON  Length

/PROB/02E1 (737)
/SOLN/0F71 (3953)

Labels:

999L    0769'        50L     00A9'        40L     0092'
200L    0176'        100L    0162'        600L    066B'
500L    063B'        230L    0143'        250L    0327'
400L    04BE'        300L    0499'        270L    03BC'
290L    0485'        460L    0616'        450L    0606'
430L    055C'        440L    05F2'        700L    06C8'
800L    0755'
```

PROHLP -- Process Help

Processing Description. This module is called whenever the user requests help by entering "HELP" or "?". It is also called to process requests for problem displays that are initiated by entering "@" or "@P". If PROHLP is being called for a HELP request, it calls PRSMEN to present the help information. If PROHLP is being called for a problem display request, it calls PRTPRB to present the problem display.

Key Local Variables. Refer to the module listing.

Module Listing. The following pages contain the FORTRAN compilation listing for PROHLP.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1             SUBROUTINE PROHLP ( IHELP, MENUID )
2       C
3       C  Subroutine PROHLP -- Process HELP
4       C
5       C  This subroutine is called by GETINT, GETFLT, or GETTXT, whenever the
6       C  user inputs "?", "HELP", "@", or "@P" in response to any prompt.
7       C  The purpose of this routine is to process the special request ac-
8       C  cordingly.  For HELP requests, PRSMEN is called to print the HELP
9       C  message.  For problem display requests, PRTPRB is called.
10      C
11      C  parameter descriptions:
12      C
13      C  IHELP  -- This item identifies the type of request as follows:
14      C                1 = HELP request
15      C                2 = Display problem on screen request
16      C                3 = Print problem request
17      C  MENUID -- This item gives the identity of the menu for which HELP
18      C             is being requested.
19      C
20      C  Check for HELP request
21      C
22            IF ( IHELP .NE. 1 ) GO TO 100
23            CALL PRSMEN ( MENUID, 1 )
24            GO TO 999
25          100 CONTINUE
26      C
27      C  Check for Display problem to screen request
28      C
29            IF ( IHELP .NE. 2 ) GO TO 200
30            CALL PRTPRB ( 5 )
31            WRITE ( 5, 902 )
32            READ ( 5,903 ) II
33            GO TO 800
34          200 CONTINUE
35            WRITE ( 5, 900 )
36            WRITE ( 5, 902 )
37            READ ( 5, 903 ) II
38            CALL PRTPRB ( 2 )
39          800 CONTINUE
40            WRITE ( 5, 901 )
41          999 CONTINUE
42          900 FORMAT ( 1H1, 10X, '>> TO PRINT THE PROBLEM <<',//,
43             1             ' ', 13X, '1.  Turn on the printer', /,
44             2             ' ', 13X, '2.  Set printer on-line', /,
45             3             ' ', 13X, '3.  Align the paper' )
46          901 FORMAT ( 1H1 )
47          902 FORMAT ( 1H0, '*** WHEN READY TO CONTINUE, PRESS THE ',
48             1             'CARRIAGE RETURN ***' )
49          903 FORMAT ( I10 )
```

```
50              RETURN
51              END

Program Unit Length=00BE (190) Bytes
Data Area Length=00E5 (229) Bytes

Subroutines Referenced:

$I0                     PRSMEN                  PRTPRB
$W2                     $ND                     $R2

Variables:

IHELP   0001*           MENUID  0003*           T:000002        0005*
II      0006*

Labels:

100L    002B'           999L    00B1'           200L    0070'
902L    009C*           903L    00E0*           800L    00A5'
900L    0008*           901L    0097*
```

PRSMEN -- Present Menu

Processing Description. PRSMEN is called to present a menu or help information to the user on the video display screen. The calling program must identify the desired menu or help information by supplying its starting menu/help file record number in the calling sequence (MENUID).

When called, PRSMEN compares MENUID to ACTMEN to determine if the requested data is already in the MENDAT common block. If it is not, PRSMEN must read the menu/help file to load it. Once the data is contained in the MENDAT common block, PRSMEN prints it to the video display screen. As specified in the calling sequence, PRSMEN will output either the menu or the help data.

Key Local Variables.

ISTART - Pointer to the character data buffer (MENVAL) for the starting character of the current line.

ISTOP -- Pointer to the character data buffer (MENVAL) for the ending character of the current line.

Module Listing. The following pages contain the FORTRAN compilation listing for PRSMEN.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE PRSMEN ( MENUID, HELPFL )
2       C
3       C  Subroutine PRSMEN -- Present menu
4       C
5       C  This subroutine is designed to retrieve and display menu or help
6       C  information to the console device.  When called PRSMEN compares
7       C  MENUID ( which indicates which menu is desired ) with ACTMEN
8       C  ( which indicates which menu is currently active ).  If they are
9       C  the same, then PRSMEN displays the menu data currently stored in
10      C  the common block /MENDAT/.  If they are different, then PRSMEN
11      C  retrieves the appropriate menu data from the menu/help file on
12      C  disk (MENDAT.DAT -- drive 2).  PRSMEN will display either menu or
13      C  help data when called, depending on the value of HELPFL.
14      C
15      C  parameter descriptions:
16      C
17      C  MENUID -- Contains the identification of the desired menu/help data.
18      C            MENUID identifies the menus starting record number in the
19      C            menu/help file.
20      C  HELPFL -- Help desired flag.  If this item is set to zero (0), then
21      C            menu data will be presented,  all other values will result
22      C            in the generation of help data.
23      C
24            LOGICAL MENVAL(768)
25            INTEGER*1 MENSIZ, MENLIN, HLPSIZ, HLPLIN, MLNSZ(12), HLNSZ(12)
26            INTEGER ACTMEN, HELPFL
27            COMMON / MENDAT / MENSIZ, MENLIN, HLPSIZ, HLPLIN, MLNSZ, HLNSZ,
28           1                  ACTMEN, MENVAL
29      C
30      C  Check ACTMEN to determine if the requested menu data is currently
31      C    active.
32      C
33            IF ( ACTMEN .EQ. MENUID ) GO TO 500
34            ACTMEN = MENUID
35          10 CONTINUE
36      C
37      C  Read the menu and help data for the desired menu from the menu/help
38      C    file and store in the common block /MENDAT/.
39      C
40            READ( 7, END=10, ERR=10, REC=MENUID ) MENSIZ, HLPSIZ, MENLIN,
41           1          HLPLIN, (MLNSZ(I),I=1,12), (HLNSZ(I),I=1,12)
42            ISTART = MENUID + 1
43            ISTOP = MENUID + HLPSIZ + MENSIZ
44            I1 = 1
45            I2 = 128
46            DO 100 I = ISTART, ISTOP
47            READ ( 7, END=10, ERR=10, REC=I )(MENVAL(J),J=I1,I2)
48            I1 = I1 + 128
49            I2 = I2 + 128
```

```
50        100 CONTINUE
51        500 CONTINUE
52      C
53      C  Determine if menu data or help data should be displayed.
54      C
55            IF ( HELPFL .NE. 0 ) GO TO 600
56            ISTART = 1
57      C
58      C  Enter loop to display the menu data to the output device.
59      C
60            DO 510 I = 1, MENLIN
61            ISTOP = ISTART + MLNSZ(I) - 1
62            WRITE ( 5, 900 ) ( MENVAL(J), J = ISTART, ISTOP )
63            ISTART = ISTOP + 1
64          510 CONTINUE
65            WRITE ( 5, 901 )
66            GO TO 9999
67          600 CONTINUE
68            ISTART = ( MENSIZ * 128 ) + 1
69            WRITE ( 5, 902 )
70      C
71      C  Enter loop to display the help data to the output device.
72      C
73            DO 610 I = 1, HLPLIN
74            ISTOP = ISTART + HLNSZ(I) - 1
75            WRITE ( 5, 900 ) ( MENVAL(J), J = ISTART, ISTOP )
76            ISTART = ISTOP + 1
77          610 CONTINUE
78            WRITE ( 5, 902 )
79            WRITE ( 5, 901 )
80          900 FORMAT ( 1H , 80A1 )
81          901 FORMAT ( 1H )
82          902 FORMAT ( 1H , '   ', 37('--') )
83         9999 CONTINUE
84            RETURN
85            END

Program Unit Length=0289 (649) Bytes
Data Area Length=004B (75) Bytes

Subroutines Referenced:

$I2                   $RS                   $ND
$W2

Variables:

MENUID  0001*         HELPFL  0003*         MENVAL  /MENDAT/+001E
MENSIZ  /MENDAT/+0000 MENLIN  /MENDAT/+0001 HLPSIZ  /MENDAT/+0002
HLPLIN  /MENDAT/+0003 MLNSZ   /MENDAT/+0004 HLNSZ   /MENDAT/+0010
ACTMEN  /MENDAT/+001C T:000002        0005*  I       000C*
```

```
T:000000        0014"   T:010000        0016"   ISTART  0018"
ISTOP   001A"           I1      001C"           I2      001E"
J       0026"

COMMON  Length

/MENDAT/031E (798)

Labels:

500L    013A'           10L     0032'           100L    012A'
600L    01D4'           510L    01B1'           900L    0028"
901L    0032"           9999L   0278'           902L    0037"
610L    024C'
```

PRTANS -- Print Answer

Processing Description. Following each problem solution, PRTANS is called to give the user the opportunity to examine problem results. Before the solution displays are presented, PRTANS informs the user if any special conditions were encountered. These special conditions are:

-- no solution
-- unbounded solution.

If an unbounded solution was detected, PRTANS will inform the user of the problem parameters that were in effect.

At this point, if there was at least one non-dominated solution, PRTANS asks the user whether he wants a summary of objective values or detailed solution displays on one or all of the solutions. Based on the user's selection, PRTANS will generate and display the data. Prior to display though, the user is asked if the display is to be generated on the video display screen or to be routed to the printer. After each display, the user is given the opportunity to get additional solution information. NOTE: Once the user exits this module, the solution data is lost.

Key Local Variables.

IDEV -- This item indicates the final destination for solution displays. It is set to 5 for the video display screen or set to 2 for the printer.

Module Listing. The following pages contain the FORTRAN compilation listing for PRTANS.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE PRTANS
2       C
3       C  Subroutine PRTANS -- Print answer
4       C
5       C  This subroutine is invoked after every problem solution.  Through
6       C  this program, the user is given the option of viewing ( either on the
7       C  screen or on the printer ) problem results.  For problems that were
8       C  solved with the weighting technique, the user is also given the
9       C  option of having displays include the objective weights the go with
10      C  the solutions.
11      C
12      C  This routine has no calling parameters
13      C
14           LOGICAL TXTVAL(32), LG(2), LX(4)
15           REAL W(4), UB(4), LB(4), DEL(4), ZF(4), OBJVAL(30,4), P(30),
16          1     VARVAL(30,10), D(10,31), CON(10,10), OBJ(4,10), RHS(10)
17           REAL SOLWHT(30,4), UBWHT(4), OBX(4)
18           REAL*8 SURP, SLACK, VV
19           REAL*8 CNAME(10), ONAME(4), FNAME
20           INTEGER IBV(10), ISTRT, NNDS
21           INTEGER*1 SPCFLG, VARNUM(30,10), CNXREF(10), JJ,
22          1        CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR, NCONX
23           COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
24          1                NOBJ, NVAR, FNAME
25           COMMON / SOLN / W, UB, LB, DEL, ISTRT, P, D, IBV, SOLWHT, VARNUM,
26          1                VARVAL, OBJVAL, ZF, NNDS, CNXREF, SPCFLG, UBWHT,
27          2                ICAP
28           DATA SURP, SLACK / 'SURPLUS ' , 'SLACK   ' /
29           DATA LX(1), LX(2), LX(3), LX(4) / 32, 32, 32, 32 /
30      C
31      C  Before printing, we must count the number of greater than constraints
32      C    and the number of equal constraints.  These counts are needed in
33      C    order to determine which variables are slacks/surpluses for which
34      C    constraints.
35      C
36           IC1 = 0
37           IC2 = 0
38           DO 5 J = 1, NCON
39           IF ( CTYPE(J) .EQ. 3 ) IC1 = IC1 + 1
40           IF ( CTYPE(J) .EQ. 2 ) IC2 = IC2 + 1
41          5 CONTINUE
42           III = ( ISTRT - 1 ) * ( NOBJ - 1 )
43      C
44      C  Set NCONX equal to the number basic variables.  NOTE:  It varies
45      C    depending on the type of solution technique.  If weighting
46      C    technique then NCONX = NCON.  If constraint technique then
47      C    NCONX = NCON + NOBJ - 1.
48      C
49           NCONX = NCON + III
```

```
50      C
51      C  Set III equal to the first variable that corresponds to the less
52      C    than constraints.  It is used to determine which variables are
53      C    slacks as follows:
54      C        If variable number <= NVAR then it is a decision variable
55      C        If variable number > NVAR but < III then it is a SURPLUS.
56      C        If variable number > III then it is a SLACK.
57      C
58            III = ( NVAR + III + IC2 + ( IC1 * 2 ) )
59          10 CONTINUE
60      C
61      C  Clear the screen
62      C
63            WRITE ( 5, 900 )
64      C
65      C  Check to see if there were no NNDS and no unbounded solutions.  If
66      C    none of either then tell the user to reformulate and try again.
67      C
68            IF ( NNDS .NE. 0 .OR. SPCFLG .NE. 0 ) GO TO 50
69            CALL GETTXT ( 219, 0, 32, 0, TXTVAL, 1 )
70            GO TO 9999
71          50 CONTINUE
72      C
73      C  Check to see if solution was not unbounded.
74      C
75            IF ( SPCFLG .EQ. 0 ) GO TO 80
76      C
77      C  Check to see if there was only one objective.
78      C
79            IF ( NOBJ .NE. 1 ) GO TO 55
80      C
81      C  Tell user that his single objective problem was unbounded.
82      C
83            CALL GETTXT ( 228, 0, 32, 0, TXTVAL, 1 )
84            GO TO 9999
85          55 CONTINUE
86      C
87      C  The following section of code displays the first weights that were
88      C    used when an unbounded solution was obtained.  NOTE:  Weights
89      C    refers to both weights and constraint limits depending on the
90      C    solution type.
91      C
92            CALL PRSMEN ( 224, 0 )
93            IF ( ISTRT .EQ. 2 ) WRITE ( 5, 902 )
94            IF ( ISTRT .EQ. 1 ) WRITE ( 5, 903 )
95            WRITE ( 5, 904 )
96            DO 60 I = ISTRT, NOBJ
97            WRITE ( 5, 905 ) I, ONAME(I), UBWHT(I)
98          60 CONTINUE
99            CALL GETTXT ( 122, 0, 32, 0, TXTVAL, 1 )
100         80 CONTINUE
```

```
C
C  Tell user how many non-dominated solutions were obtained.
C
      WRITE ( 5, 901 ) NNDS
      IF ( NNDS .NE. 0 ) GO TO 100
      CALL GETTXT ( 122, 0, 32, 0, TXTVAL, 1 )
      GO TO 9999
  100 CONTINUE
C
C  If more then the upper limit of 30 solutions was obtained, inform the
C    user of that fact.
C
      IF ( ICAP .NE. 0 ) CALL PRSMEN ( 252, 0 )
C
C  Ask the user which type of solution display he desires.
C      1 -- objective summary display
C      2 -- detailed solution display, one solution
C      3 -- detailed solution display, all solution
C      0 -- EXIT display module
C
      CALL GETINT ( 233, 0, 3, 0, ITYPE )
      GO TO ( 110, 105, 102 ), ITYPE
      GO TO 9999
  102 CONTINUE
C
C  User has requested to see details on all solutions.  Set display
C    loop limits.
C
      I1 = 1
      I2 = NNDS
      GO TO 109
  105 CONTINUE
C
C  User has requested details on just one solution, ask him which one.
C
      CALL GETINT ( 214, 0, NNDS, 1, I1 )
      I2 = I1
  108 CONTINUE
      IWGHT = 2
C
C  If this was a weighting technique solution, ask user if he would like
C    his displays to show representative objective weights.
C
      IF ( ISTRT .EQ. 2 ) GO TO 110
      CALL GETINT ( 239, 0, 2, 1, IWGHT )
  110 CONTINUE
C
C  Ask user where to generate the requested solution display
C      1 -- Display screen
C      2 -- Printer
C
```

```
      CALL GETINT ( 244, 0, 2, 1, IDEV )
      WRITE ( 5, 900 )
      IF ( IDEV .EQ. 1 ) GO TO 200
C
C  User has requested printer.  Give him instructions and wait for ack-
C    nowledgement.
C
      CALL GETTXT ( 248, 0, 32, 0, TXTVAL, 1 )
      GO TO 210
  200 CONTINUE
      IDEV = 5
  210 CONTINUE
      IF ( ITYPE .GE. 2 ) GO TO 240
C
C  Begin code for generating the objective value summary.
C
      WRITE ( IDEV, 908 ) ( ONAME(I), I = 1, NOBJ )
      WRITE ( IDEV, 913 ) ( LX(I), I = 1, NOBJ )
      DO 230 I = 1, NNDS
      DO 220 J = 1, NOBJ
      OBX(J) = OBJVAL(I,J)
  220 CONTINUE
      WRITE ( IDEV, 917 ) I, ( OBX(J), J = 1, NOBJ )
  230 CONTINUE
      IF ( IDEV .EQ. 2 ) GO TO 810
      WRITE ( 5, 919 )
      READ ( 5, 915 ) II
      WRITE ( 5, 900 )
      GO TO 10
  240 CONTINUE
C
C  Begin code to generate detailed solution displays.
C
      WRITE ( IDEV, 906 ) FNAME
C
C  Enter loop to process the requested number of solutions
C    (ie all or one in particular)
C
      DO 800 I = I1, I2
      WRITE ( IDEV, 907 ) I
C
C  Enter loop to output objective values.
C
      DO 300 J = 1, NOBJ
      OB = OBJVAL(I,J)
      IF ( IWGHT .EQ. 1 ) GO TO 250
      WRITE ( IDEV, 909 ) J, ONAME(J), OB
      GO TO 300
  250 CONTINUE
      WRITE ( IDEV, 910 ) J, ONAME(J), OB, SOLWHT(I,J)
  300 CONTINUE
```

```
C
C  The following loop is designed to allow processing of the three
C     types of variables ( decision, slack, surplus ) in order.
C
      DO 600 K = 1, 3
      IF ( K .GT. 1 ) GO TO 380
      WRITE ( IDEV, 911 )
      GO TO 400
  380 CONTINUE
      IF ( K .GT. 2 ) GO TO 400
      WRITE ( IDEV, 916 )
  400 CONTINUE
C
C  Enter loop to search of of the basic variables for one of the 3
C     types ( depending on the value of loop variable K ).
C
      DO 500 J = 1, NCONX
      AX = VARVAL(I,J)
C
C  Don't display uninteresting basic varibales that are equal to 0.
C
      IF ( AX .EQ. 0.0 ) GO TO 500
      JJ = VARNUM(I,J)
      IF ( K .GT. 1 ) GO TO 450
C
C  This section processes decision variables ( K = 1 ).
C
      IF ( JJ .GT. NVAR ) GO TO 500
      ENCODE ( LG, 915 ) JJ
      IF ( JJ .GE. 10 ) GO TO 420
      LG(1) = LG(2)
      LG(2) = 32
  420 CONTINUE
      WRITE ( IDEV, 912 ) LG(1), LG(2), AX
      GO TO 500
  450 CONTINUE
      II = JJ - NVAR
      IF ( K .GT. 2 ) GO TO 460
C
C  This section processes SURPLUS variables ( K = 2 ).
C
      IF ( JJ .GT. IIY .OR. JJ .LE. NVAR ) GO TO 500
      VV = SURP
      GO TO 480
  460 CONTINUE
C
C  This section processes SLACK variables ( K = 3 ).
C
      IF ( JJ .LE. III ) GO TO 500
      II = II - IC1
      VV = SLACK
```

```
  480 CONTINUE
      II = CNXREF(II)
      WRITE ( IDEV, 914 ) II, CNAME(II), VV, AX
  500 CONTINUE
  600 CONTINUE
  610 CONTINUE
      IF ( IDEV .NE. 2 ) GO TO 650
C
C  For printer outputs, do a page eject.
C
      WRITE ( 2, 918 )
      GO TO 800
  650 CONTINUE
C
C  For screen outputs, ask user for input before going on.
C
      WRITE ( 5, 919 )
      READ ( 5, 915 ) II
      WRITE ( 5, 900 )
  800 CONTINUE
  810 CONTINUE
      IF ( IDEV .EQ. 2 ) WRITE ( 2, 900 )
      GO TO 10
  900 FORMAT ( 1H1 )
  901 FORMAT ( 1H1, 'There were ', I2, ' NON-DOMINATED solutions ',
     1            'found.', / )
  902 FORMAT ( 1H0, 14X, 'OBJECTIVE', 9X, 'LIMIT' )
  903 FORMAT ( 1H0, 14X, 'OBJECTIVE', 9X, 'WEIGHT' )
  904 FORMAT ( 1H , 13X, '-----------', 7X, '--------' )
  905 FORMAT ( 1H , 13X, I1, '. ', A8, 7X, F8.2 )
  906 FORMAT ( 1H , 18X, 'NON-DOMINATED SOLUTIONS FOR PROBLEM: ', A8 )
  907 FORMAT ( 1H0, 30X, '*** SOLUTION ', I2, ' ***', //,
     1            ' ', 29X, '----- OBJECTIVES -----' )
  908 FORMAT ( 1H , 10X, 'SUMMARY OF OBJECTIVE VALUES FOR THE NON-',
     1            'DOMINATED SOLUTIONS', //, ' ', 15X,
     2            'OBJECTIVES -----', /, ' ', 2X, 'SOLUTION',
     3            4(7X,A8) )
  909 FORMAT ( 1H , 26X, I2, '.  ', A8, ' = ', F12.4 )
  910 FORMAT ( 1H , 13X, I2, '.  ', A8, ' = ', F12.4, 5X, '( WEIGHT = ',
     1            F7.5, ' )' )
  911 FORMAT ( 1H0, 29X, '----- VARIABLES -----' )
  912 FORMAT ( 1H , 22X, 'Decision variable x', 2A1, ' = ', F12.4 )
  913 FORMAT ( 1H , 2X, '--------', 2X, 4(2X,A1,'----------') )
  914 FORMAT ( 1H , 12X, 'Constraint', I2, ' ( ', A8,
     1            ' ) had a ', A8, '= ', F12.4 )
  915 FORMAT ( I2 )
  916 FORMAT ( 1H0, 20X, '----- CONSTRAINT SLACKS/SURPLUSES -----' )
  917 FORMAT ( 1H , 5X, I2, 5X, 4(3X,F12.4) )
  918 FORMAT ( 1H , 20('----') )
  919 FORMAT ( 1H0, 20X, '--- PRESS THE RETURN KEY TO CONTINUE ---' )
 9999 CONTINUE
```

```
305           RETURN
306           END

Program Unit Length=0868 (2152) Bytes
Data Area Length=042B (1067) Bytes

Subroutines Referenced:

$I3                   $I2                   $I1
$I0                   $M9                   $W2
$ND                   GETTXT                PRSMEN
GETINT                $CG                   $L1
$T1                   $MA                   $NB
$R2                   $OE                   $L3
$T3

Variables:

TXTVAL  0001"         LG      0021"         LX      0023"
W       /SOLN/+0000   UB      /SOLN/+0010   LB      /SOLN/+0020
DEL     /SOLN/+0030   ZF      /SOLN/+0F42   OBJVAL  /SOLN/+0D62
P       /SOLN/+0042   VARVAL  /SOLN/+08B2   D       /SOLN/+00BA
CON     /PROB/+0000   OBJ     /PROB/+0212   RHS     /PROB/+01EA
SOLWHT  /SOLN/+05A6   USWHT   /SOLN/+0F5F   OBX     0027"
SURP    0037"         SLACK   003F"         VV      0047"
CNAME   /PROB/+0190   ONAME   /PROB/+02B2   FNAME   /PROB/+02D9
IBV     /SOLN/+0592   ISTRT   /SOLN/+0040   NNDS    /SOLN/+0F52
SPCFLG  /SOLN/+0F5E   VARNUM  /SOLN/+0786   CNXREF  /SOLN/+0F54
JJ      004F"         CTYPE   /PROB/+01E0   OTYPE   /PROB/+02D2
NCON    /PROB/+02D6   NOBJ    /PROB/+02D7   NVAR    /PROB/+02D8
NCONX   0050"         ICAP    /SOLN/+0F6F   IC1     0051"
IC2     0053"         J       0055"         T:000002      0057"
III     0058"         T:010002      005A"   I       006B"
T:000000      006D"   T:010000      006F"   ITYPE   0081"
II      0089"         I2      008B"         IWGHT   0093"
IDEV    009B"         II      00AB"         OB      00AD"
T:020000      00B1"   K       00B3"         AX      00B5"

COMMON  Length

/PROB/02E1 (737)
/SOLN/0F71 (3953)

Labels:

5L      0048'         10L     009C'         900L    00B9'
50L     00D7'         9999L   0827'         80L     01C5'
55L     0101'         902L    00FB"         903L    011B"
904L    013C"         60L     01A5'         905L    0161"
901L    00BE"         100L    01FE'         110L    027E'
105L    0245'         102L    0236'         108L    0257'
```

200L	02B9'	210L	02BF'	240L	043F'
908L	01F8"	913L	032A"	230L	03E8'
220L	038B'	917L	03CD"	810L	0803'
919L	03F7"	915L	0396"	906L	017D"
800L	07F3'	907L	01B1"	300L	0554'
250L	04EF'	909L	027D"	910L	029E"
600L	0793'	380L	0592'	911L	02DA"
400L	05B4'	916L	039A"	500L	077F'
450L	068D'	420L	065D'	912L	02FB"
460L	06F4'	480L	0732'	914L	0358"
610L	07A0'	650L	07C4'	918L	03E7"

PRTPRB -- Print Problem

Processing Description. PRTPRB is called for the purpose of generating a display of the current problem whenever the user enters "@" or "@P". The added "P" is used to indicate that the display should be routed to the printer rather than the video display screen.

The display format is the same for both the printer and the screen. PRTPRB extracts all the necessary problem information from the problem common block (PROB).

Key Local Variables. Refer to the module listing.

Module Listing. The following pages contain the FORTRAN compilation listing for PRTPRB.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE PRTPRB ( IDEV )
2       C
3       C  Subroutine PRTPRB -- Print Problem
4       C
5       C  This subroutine is invoked whenever the user inputs "@" or "@P" in
6       C  response to any PCADA prompt.  It is designed to output the contents
7       C  of the common block / PROB / to the line printer ( for "@P" ) or to
8       C  the screen ( for "@" ).  The same format is used for both outputs.
9       C  The only distinguishing feature between the to types of outputs is
10      C  the device ( IDEV ).
11      C
12      C  parameter description --
13      C
14      C  IDEV -- This item identifies the device to which the output is to be
15      C          directed.
16      C
17           LOGICAL LINE1(23), LINE2(48), LINE3(14), LINE4(13)
18           REAL OBT(3)
19           REAL CON(10,10), OBJ(4,10), RHS(10)
20           REAL*8 CNAME(10), ONAME(4), FNAME
21           INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
22           COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
23          1                NOBJ, NVAR, FNAME
24           DATA OBT / 'MAXI', 'MINI', 'MIZE' /
25      C
26      C  If the display is going to the screen, clear it.
27      C
28           IF ( IDEV .EQ. 5 ) WRITE ( 5, 906 )
29      C
30      C  Write problem name and header
31      C
32           WRITE ( IDEV, 900 ) FNAME
33      C
34      C  Enter loop to write out the objectives
35      C  NOTE:  The output of the objectives uses 2 items: LINE1 and LINE2.
36      C    LINE1 contains the objective number, objective name, and
37      C    objective type.  LINE1 is output only on the first line for the
38      C    objective.  LINE2 contains ANSII dat for up to 4 variables, it is
39      C    set by repeated calls to CDELIN. After the first line, LINE1 is
40      C    blanked out.
41      C
42           DO 10 I = 1, 48
43           LINE2(I) = 32
44          10 CONTINUE
45           DO 40 I = 1, NOBJ
46           II = OTYPE(I)
47           ENCODE ( LINE1, 901 ) I, OBT(II), OBT(3), ONAME(I)
48           II = 1
49           DO 30 J = 1, NVAR
```

```
50          CALL CDELIN ( OBJ(I,J), LINE2, II, J )
51      C
52      C  Determine if a line should be printed.
53      C
54          IF ( J .NE. NVAR .AND. J .NE. 4 .AND. J .NE. 8 ) GO TO 30
55          WRITE ( IDEV, 902 ) ( LINE1(K), K = 1, 23 ),
56         1                    ( LINE2(L), L = 1, 48 )
57      C
58      C  Enter loop to blank out LINE1 and LINE2.  NOTE: It is ESSENTIAL
59      C    that LINE1 and LINE2 physically reside sequentially since the
60      C    following single loop is used to blank out both of them.
61      C
62          DO 25 K = 1, 71
63          LINE1(K) = 32
64        25 CONTINUE
65          II = 1
66        30 CONTINUE
67          WRITE ( IDEV, 907 )
68        40 CONTINUE
69      C
70      C  Write out the constraint header
71      C
72          WRITE ( IDEV, 903 )
73      C
74      C  Enter loop to output the constraints.
75      C  NOTE: This output uses LINE3, LINE2, and LINE4.  LINE2 is used as
76      C    noted above, LINE3 is used only for the first line of each con-
77      C    straint and contains the constraint number and name.  LINE4 is
78      C    used only for the last line of the constraint and contains the
79      C    constraint sign and the right hand side value.
80      C
81          DO 100 I = 1, NCON
82      C
83      C  Enter loop to blank out LINE4
84      C
85          DO 45 K = 1, 13
86          LINE4(K) = 32
87        45 CONTINUE
88          ENCODE ( LINE3, 904 ) I, CNAME(I)
89          II = 1
90      C
91      C  Enter loop to process each of the variables.
92      C
93          DO 60 J = 1, NVAR
94          CALL CDELIN ( CON(I,J), LINE2, II, J )
95          IF ( J .NE. NVAR .AND. J .NE. 4 .AND. J .NE. 8 ) GO TO 60
96          IF ( J .NE. NVAR ) GO TO 50
97          ENCODE ( LINE4, 905 ) RHS(I)
98          LINE4(3) = 61
99          IF ( CTYPE(I) .EQ. 2 ) GO TO 50
100         LINE4(2) = CTYPE(I) + 59
```

```
101        50 CONTINUE
102           WRITE ( IDEV, 902 ) ( LINE3(K), K = 1, 14 ),
103          1                    ( LINE2(L), L = 1, 48 ),
104          2                    ( LINE4(M), M = 1, 13 )
105      C
106      C  Enter loop to blank out LINE2 and LINE3.  As above, they must re-
107      C    side sequentially.
108      C
109           DO 55 K = 1, 62
110           LINE2(K) = 32
111        55 CONTINUE
112           II = 1
113        60 CONTINUE
114           WRITE ( IDEV, 907 )
115       100 CONTINUE
116      C
117      C  If output is going to the printer, do a form feed.
118      C
119           IF ( IDEV .EQ. 2 ) WRITE ( 2, 906 )
120       900 FORMAT ( 1H ,'PROBLEM NAME = ', A8, //,
121          1            ' ----- OBJECTIVES -----', / )
122       901 FORMAT ( I2, '. ', 2A4, ' ', A8, '= ' )
123       902 FORMAT ( 1H , 75A1 )
124       903 FORMAT ( 1H0,  '----- SUBJECT TO THE FOLLOWING CONSTRAINTS',
125          1               ' -----', / )
126       904 FORMAT ( I2, '. ',A8, ': ' )
127       905 FORMAT ( '     ', F8.2 )
128       906 FORMAT ( 1H1 )
129       907 FORMAT ( 1H  )
130           RETURN
131           END
```

Program Unit Length=0474 (1140) Bytes
Data Area Length=0150 (336) Bytes

Subroutines Referenced:

$I3	$I2	$I1
$I0	$W2	$ND
$OE	CDELIN	$M9

Variables:

IDEV	0001"	LINE1	0003"	LINE2	001A"
LINE3	004A"	LINE4	0058"	OBT	0065"
CON	/PROB/+0000	OBJ	/PROB/+0212	RHS	/PROB/+01EA
CNAME	/PROB/+0190	ONAME	/PROB/+02B2	FNAME	/PROB/+02D9
CTYPE	/PROB/+01E0	OTYPE	/PROB/+02D2	NCON	/PROB/+02D6
NOBJ	/PROB/+02D7	NVAR	/PROB/+02D8	T:000002	0071"
I	0072"	T:000000	0074"	II	0076"
T:010000	0078"	J	007A"	T:010002	0080"

```
T:020002        0081"   K       0082"           L       0084"
M       008A"           T:020000        008C"

COMMON  Length

/PROB/02E1 (737)

Labels:

906L    0146"           900L    008E"           10L     0057'
40L     01F5'           901L    00C7"           30L     0105'
902L    00E0"           25L     01C2'           907L    014B"
903L    00EA"           100L    042E'           45L     0231'
904L    0127"           60L     040E'           50L     035E'
905L    0138"           55L     03FB'
```

SAVPRB -- Save Problem

Processing Description. SAVPRB is called to save the problem file that is in the problem common block (PROB) to disk. It is assumed that the disk file has already been opened. Problem information is saved to the disk problem file in the format as specified in chapter II. If SAVPRB encounters some type of input/output error, it sets the status flag (ISTAT) to 1.

Key Local Variables.

IREC -- This item is used as a pointer to the problem disk file record that is being written to.

Module Listing. The following pages contain the FORTRAN compilation listing for SAVPRB.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE SAVPRB ( ISTAT )
2       C
3       C  Subroutine SAVPRB -- Save problem
4       C
5       C  This subroutine is designed to save the current problem, as contained
6       C  in the common block /PROB/ to the diskfile identified by FNAME.
7       C  It is invoked automatically following problem definition or following
8       C  problem editing.
9       C
10      C  Parameter description ----
11      C
12      C  ISTAT -- This item is returned to the calling program to indicate the
13      C           status of the save action.  A zero value indicates that the
14      C           problem was successfully saved.  A non-zero value indicates
15      C           that some problem was encountered.
16      C
17           INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
18           REAL CON(10,10), OBJ(4,10), RHS(10)
19           REAL*8 CNAME(10), ONAME(4), FNAME
20           COMMON /PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
21          1               NOBJ, NVAR, FNAME
22      C
23      C  Set the status flag to indicate no problem
24      C
25           ISTAT = 0
26      C
27      C  write the controlling data to the first record.  note: if an i/o
28      C    error occurs, control will revert to statement 10
29      C
30           WRITE ( 8, ERR=10, END=10, REC=1 ) NCON, NOBJ, NVAR
31           GO TO 20
32          10 CONTINUE
33      C
34      C  set status flag to indicate that an I/O error has occured.
35      C
36           ISTAT = 1
37           GO TO 9999
38          20 CONTINUE
39      C
40      C  loop on the number of objectives to write the objective data to the
41      C    file to be saved.
42      C
43           DO 30 I = 1, NOBJ
44           IREC = I + 1
45           WRITE ( 8, ERR=10, END=10, REC=IREC ) ONAME(I), OTYPE(I),
46          1           ( OBJ(I,J), J = 1, NVAR )
47          30 CONTINUE
48      C
49      C  loop on the number of constraints to save the constraint data to the
```

```
50      C    file to be saved.
51      C
52           DO 40 I = 1, NCON
53           IREC = I + 1 + NOBJ
54           WRITE ( 8, ERR=10, END=10, REC=IREC ) CNAME(I), CTYPE(I),
55          1           ( CON(I,J), J = 1, NVAR ), RHS(I)
56        40 CONTINUE
57      9999 CONTINUE
58           ENDFILE 8
59           RETURN
60           END

Program Unit Length=01AF (431) Bytes
Data Area Length=0029 (41) Bytes

Subroutines Referenced:

$I3                   $I2                   $I1
$W5                   $ND                   $M9
$EN

Variables:

ISTAT   0001"          CTYPE   /PROB/+01E0     OTYPE   /PROB/+02D2
NCON    /PROB/+02D6    NOBJ    /PROB/+02D7     NVAR    /PROB/+02D8
CON     /PROB/+0000    OBJ     /PROB/+0212     RHS     /PROB/+01EA
CNAME   /PROB/+0190    ONAME   /PROB/+02B2     FNAME   /PROB/+02D9
I       000D"          IREC    000F"           J       0017"
T:000000       0019"   T:010000       001B"   T:020000       001D"
T:030000       001F"   T:040000       0027"

COMMON  Length

/PROB/02E1 (737)

Labels:

10L     002D'          20L     003A'          9999L   019C'
30L     00C6'          40L     0188'
```

SOLVE -- Solve Problem

Processing Description. This subroutine is called by PRBSLV after the user has specified all solution parameters. SOLVE repeatedly calls CSOLVE or WSOLVE to process each iteration. The purpose of SOLVE is to determine the "weights" to be used for the next iteration. NOTE: SOLVE works the same for both the weighting and constraint techniques. That is, the weights are incremented using the user supplied increment between the user supplied lower and upper limits.

Key Local Variables. Refer to the module listing.

Module Listing. The following pages contain the FORTRAN compilation listing for SOLVE.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE SOLVE ( ITER )
2       C
3       C  Subroutine SOLVE -- Solve problem
4       C
5       C  This program is called by PRBSLV to further process the problem
6       C  solution.  It does so by iterating on the objective weights ( for the
7       C  weigting technique ) or the constraint limits ( for the constraint
8       C  technique ), and calling WSOLVE/CSOLVE for each legitimate problem.
9       C
10      C  SOLVE does not use any calling parameters.
11      C
12           REAL P(30), D(10,31)
13           REAL SOLWHT(30,4), UBWHT(4)
14           REAL ZF(4), OBJVAL(30,4), VARVAL(30,10)
15           INTEGER IBV(10)
16           REAL W(4), UB(4), LB(4), DEL(4), CON(10,10), OBJ(4,10), RHS(10)
17           REAL*8 CNAME(10), ONAME(4), FNAME
18           INTEGER*1 SPCFLG, VARNUM(30,10), CNXREF(10)
19           INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
20           COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
21          1                NOBJ, NVAR, FNAME
22           COMMON / SOLN / W, UB, LB, DEL, ISTRT, P, D, IBV, SOLWHT, VARNUM,
23          1                VARVAL, OBJVAL, ZF, NNDS, CNXREF, SPCFLG, UBWHT,
24          2                ICAP
25      C
26      C  for starters, set the "weights" equal to the lower limit.  NOTE:
27      C    the weight array W will contain weights for the weighting tech-
28      C    nique and right hand side values for the constraint technique.  The
29      C    processing by this program is the same for both.  The "weight"
30      C    simply assumes values begining at the lower bound and incrementing
31      C    by DEL until the upper bound is reached.
32      C
33           DO 100 I = 1, NOBJ
34           W(I) = LB(I)
35         100 CONTINUE
36           WRITE ( 5, 900 ) ITER
37           II = 1
38      C
39      C  enter loop to process every possible weight/constraint combination.
40      C
41           DO 400 I = 1, 20000
42      C
43      C  for constraint technique must jump to insure that we process the
44      C    case with all constraints set to the lower bound.
45      C
46           IF ( I .EQ. 1 .AND. ISTRT .EQ. 2 ) GO TO 350
47      C
48      C  enter loop to increment the weights.
49      C
```

```
50          DO 300 J = ISTRT, NOBJ
51          IF ( DEL(J) .EQ. 0.0 ) GO TO 200
52          W(J) = W(J) + DEL(J)
53          IF ( W(J) .LT. ( UB(J) + 0.001 ) ) GO TO 350
54          W(J) = LB(J)
55        200 CONTINUE
56          IF ( J .EQ. NOBJ ) GO TO 9999
57        300 CONTINUE
58        350 CONTINUE
59      C
60      C  check to see if this request is for the constraint technique-ifso
61      C    all that needs be done is to call CSOLVE
62      C
63          IF ( ISTRT .NE. 2 ) GO TO 360
64          WRITE ( 5, 901 ) II
65          CALL CSOLVE
66          GO TO 390
67        360 CONTINUE
68      C
69      C  for weighting technique requests, we must insure that the weights sum
70      C    to one, ifso, fine-call WSOLVE, otherwise, throw out that set of
71      C    weights and go back to get the next set.
72      C
73          SUM = 0.0
74          DO 380 J = 1, NOBJ
75          SUM = SUM + W(J)
76        380 CONTINUE
77          IF ( SUM .GT. 1.001 .OR. SUM .LT. 0.999 ) GO TO 400
78          WRITE ( 5, 901 ) II
79          CALL WSOLVE
80        390 CONTINUE
81          IF ( ICAP .NE. 0 ) GO TO 9999
82          II = II + 1
83        400 CONTINUE
84        900 FORMAT ( 1H1, ///, ' ', 13X, '---> MULTIOBJECTIVE PROBLEM ',
85           1         'SOLUTION IN PROCESS <---', ///, ' ', 30X, 'PLEASE BE',
86           2       ' PATIENT!', //, ' ', 5X, 'As defined, this problem will ',
87           3         'require ', I4, ' linear program iterations.', //,
88           4         25X, 'CURRENTLY WORKING ITERATION:', // )
89        901 FORMAT ( 1H+, 34X, '*** ', I4, ' ***' )
90       9999 CONTINUE
91          RETURN
92          END

Program Unit Length=023F (575) Bytes
Data Area Length=0124 (292) Bytes

Subroutines Referenced:

$IO                 $L1                 $T1
$W2                 $ND                 $AB
```

```
$SB                    CSOLVE                 WSOLVE

Variables:

ITER    0001"          P       /SOLN/+0042    D       /SOLN/+00BA
SOLWHT  /SOLN/+05A6    UBWHT   /SOLN/+0F5F    ZF      /SOLN/+0F42
OBJVAL  /SOLN/+0D62    VARVAL  /SOLN/+08B2    IBV     /SOLN/+0592
W       /SOLN/+0000    UB      /SOLN/+0010    LB      /SOLN/+0020
DEL     /SOLN/+0030    CON     /PROB/+0000    OBJ     /PROB/+0212
RHS     /PROB/+01EA    CNAME   /PROB/+0190    ONAME   /PROB/+02B2
FNAME   /PROB/+02D9    SPCFLG  /SOLN/+0F5E    VARNUM  /SOLN/+0786
CNXREF  /SOLN/+0F54    CTYPE   /PROB/+01E0    OTYPE   /PROB/+02D2
NCON    /PROB/+02D6    NOBJ    /PROB/+02D7    NVAR    /PROB/+02D8
ISTRT   /SOLN/+0040    NNDS    /SOLN/+0F52    ICAP    /SOLN/+0F6F
I       0003"          T:000000       0005"   T:010000       0007"
II      0009"          T:000002       000B"   T:010002       000C"
J       000D"          T:000001       000F"   SUM    0013"

COMMON  Length

/PROB/02E1 (737)
/SOLN/0F71 (3953)

Labels:

100L    0028'          900L    0017"          400L    0219'
350L    0149'          300L    0135'          200L    0114'
9999L   0226'          360L    017B'          901L    010A"
390L    01FF'          380L    01A5'
```

WSOLVE -- Solve Using Weighting Technique

Processing Description. This module is responsible for calling the linear programming subroutine (LP) to obtain a problem solution using the weighting technique. Prior to invoking WSOLVE, SOLVE has determined from the user the desired objective weights and increments. WSOLVE then sets up the problem in the format required by LP, and then calls LP for solution.

Following the call to the linear programming package, PROANS is called to examine the LP solution to determine if it should be included in the non-dominated solution set. WSOLVE is called once by SOLVE for each iteration. The number of iterations is determined by the weights and weight increments as specified by the user.

Key Local Variables.

IMULT -- Flag that is used to convert objective coefficients when the objective is to be minimized. That is, since LP only maximizes, for objectives to be minimized, the coefficients need to be multiplied by minus 1. For minimize problems, IMULT will assume the value of -1, for maximize problems, it will be set to 1.

Module Listing. The following pages contain the FORTRAN compilation listing of WSOLVE.

```
FORTRAN-80 Ver. 3.43 Copyright 1978-1981 (C) By Microsoft -- Bytes: 25131
Created: 27-Jun-81
1            SUBROUTINE WSOLVE
2       C
3       C  Subroutine WSOLVE -- Solve problem with the weighting technique
4       C
5       C  This subroutine is called by SOLVE whenever a problem is to be
6       C  solved with the weighting technique.  Its purpose is to iteratively
7       C  call LP with different weights as specified by the user.  It does so
8       C  by formatting the problem and moving its description from / PROB /
9       C  to / SOLN /.  After calling LP, this subroutine also examines the
10      C  results and saves the nondominated solutions for later processing.
11      C
12      C  This subroutine has no calling parameters.
13      C
14           REAL W(4), UB(4), LB(4), DEL(4), CON(10,10), OBJ(4,10), RHS(10)
15           REAL SOLWHT(30,4), UBWHT(4)
16           REAL P(30), D(10,31)
17           REAL ZF(4), OBJVAL(30,4), VARVAL(30,10)
18           INTEGER IMULT(4), IBV(10)
19           REAL*8 CNAME(10), ONAME(4), FNAME
20           INTEGER*1 SPCFLG, VARNUM(30,10), CNXREF(10)
21           INTEGER*1 CTYPE(10), OTYPE(4), NCON, NOBJ, NVAR
22           COMMON / PROB / CON, CNAME, CTYPE, RHS, OBJ, ONAME, OTYPE, NCON,
23          1                NOBJ, NVAR, FNAME
24           COMMON / SOLN / W, UB, LB, DEL, ISTRT, P, D, IBV, SOLWHT, VARNUM,
25          1                VARVAL, OBJVAL, ZF, NNDS, CNXREF, SPCFLG, UBWHT,
26          2                ICAP
27      C
28      C  Enter loop to set the multipliers.  Since LP processes only MAXIMIZE
29      C    problems, MINIMIZE objectives must be converted by multiplying them
30      C    by -1.  MAXIMIZE objectives are OK as is and thus are multiplied
31      C    1.
32      C
33           DO 100 I = 1, NOBJ
34           IMULT(I) = 1
35           IF ( OTYPE(I) .EQ. 2 ) IMULT(I) = -1
36         100 CONTINUE
37      C
38      C  Enter loop to calculate the weighted objective coefficients and store
39      C    them in P for processing by LP.
40      C
41           DO 200 I = 1, NVAR
42           P(I) = 0.0
43           DO 150 J = 1, NOBJ
44           P(I) = P(I) + W(J) * OBJ(J,I) * IMULT(J)
45         150 CONTINUE
46         200 CONTINUE
47      C
48      C  set the pointer to the SOLN constraint array to its initial condition
49      C
```

```
50            II = 1
51      C
52      C  call CONMVE to move "greater than or equal" constraints from CON
53      C    ( in PROB ) to d ( in SOLN ).
54      C
55            CALL CONMVE ( 3, IA, II )
56      C
57      C  call CONMVE to move "equal" constraints from CON ( in PROB ) to D
58      C    ( in SOLN ).
59      C
60            CALL CONMVE ( 2, NEQ, II )
61      C
62      C  call CONMVE to move "less than or equal" constraints from CON
63      C    ( in PROB ) to D ( in SOLN ).
64      C
65            CALL CONMVE ( 1, IX, II )
66      C
67      C  set the number of constraints
68      C
69            IW = NCON
70      C
71      C  set the number of variables ( plus one )
72      C
73            IY = NVAR + 1
74      C
75      C  set the number of matrix columns ( including one for the right hand
76      C    side.
77      C
78            IZ = IY + NEQ + IX + ( 2 * IA )
79      C
80      C  call LP to get the solution.
81      C
82            CALL LP ( IW, IZ, IY, IA, NEQ )
83            CALL PROANS ( IZ, IW )
84            RETURN
85            END

Program Unit Length=0174 (372) Bytes
Data Area Length=002A (42) Bytes

Subroutines Referenced:

$L1                   $T1                   $CA
$MB                   $AB                   CONMVE
LP                    PROANS

Variables:

W       /SOLN/+0000     UB      /SOLN/+0010     LB      /SOLN/+0020
DEL     /SOLN/+0030     CON     /PROB/+0000     OBJ     /PROB/+0212
RHS     /PROB/+01EA     SOLWHT  /SOLN/+05A6     UBWHT   /SOLN/+0F5F
```

```
P       /SOLN/+0042     D       /SOLN/+008A     ZF      /SOLN/+0F42
OBJVAL  /SOLN/+0D62     VARVAL  /SOLN/+08B2     IMULT   0001"
IBV     /SOLN/+0592     CNAME   /PROB/+0190     ONAME   /PROB/+02B2
FNAME   /PROB/+02D9     SPCFLG  /SOLN/+0F5E     VARNUM  /SOLN/+0786
CNXREF  /SOLN/+0F54     CTYPE   /PROB/+01E0     OTYPE   /PROB/+02D2
NCON    /PROB/+02D6     NOBJ    /PROB/+02D7     NVAR    /PROB/+02D8
ISTRT   /SOLN/+0040     NNDS    /SOLN/+0F52     ICAP    /SOLN/+0F6F
I       0009"           T:000000        0008"   T:000002        000D"
J       000E"           T:010000        0010'   T:020000        0012"
T:030000        0014"   II      0016"           IA      0018"
NE@     001A"           IX      001C"           IW      001E"
IY      0020"           IZ      0022"

COMMON  Length

/PROB/02E1 (737)
/SOLN/0F71 (3953)

Labels:

100L    0044'           200L    00E5'           150L    00D1'
```

VI Menu/Help File Data

The following pages contain a listing of the actual menu and help data as stored in the menu/help file. The structure of the menu/help file is described in Chapter II.

Menus are referenced by the various PCADA software modules using the menu's starting record number. In general, all modules that need to ask the user a question will call one of the input processors (GETFLT, GETINT, or GETTXT) with the desired menu's starting record number in the subroutine calling sequence.

Some subroutines display information or messages to the user that are not in the form of a question or menu that requires a response from the user. These displays are also contained in the menu/help file and are distinguished from the others with a special HELP text that says: "HELP not applicable for this display...". For an example of this type of menu/help file entry, refer to menu #1.

A few PCADA questions contain variable data. These menus are generated by the software in real time. However, these menus still utilize the menu/help file in order to contain HELP data for the question. For an example of this type of menu/help file entry, refer to menu #14.

```
MENU NUMBER  1  STARTING RECORD NUMBER     2

     ----- MENU TEXT -----

          ************************************************
          *                                              *
          *  PERSONAL COMPUTER AIDED DECISION ANALYSIS   *
          *                                              *
          *                  (PCADA)                     *
          *                                              *
          ************************************************

This program is designed to simplify the decision making process.  It does so
by providing an easy to use tool for solving multiple objective problems.

     ----- HELP TEXT -----

HELP not applicable for this display...

--------------------------------------------------------------------------------

MENU NUMBER  2  STARTING RECORD NUMBER     9

     ----- MENU TEXT -----

                         ---- GENERAL INSTRUCTIONS ----

Although most program prompts are self-explanatory, you may receive clarifying
information to any question by responding to it with "HELP" or "?".  Once you
have defined a problem or loaded one from disk, you may have it displayed on the
screen (or at the printer) by entering "@" (for printer enter "@P") in response
to any question or menu.  Before continuing, please insure that you have an
initialized diskette for problem files installed in disk drive 2.

     ----- HELP TEXT -----

HELP not applicable for this display...

--------------------------------------------------------------------------------

MENU NUMBER  3  STARTING RECORD NUMBER    15

    ----- MENU TEXT -----

         WHEN READY TO BEGIN, PRESS THE CARRIAGE RETURN BUTTON (CR)

     ----- HELP TEXT -----

To begin PCADA processing you should now press the carriage return button.

--------------------------------------------------------------------------------
```

MENU NUMBER 4 STARTING RECORD NUMBER 18

----- MENU TEXT -----

PERSONAL COMPUTER AIDED DECISION ANALYSIS
(PCADA)

Problems that you defined this session have been saved to disk. Your problem files can be identified on the problem diskette as those of type "PRB". You should take care to manage these files by deleting or re-defining those that are obsolete. Also, please make note of your problem file names for future use.

---- END PROGRAM ----

----- HELP TEXT -----

HELP not applicable for this display ....

MENU NUMBER 5 STARTI..G RECORD NUMBER 24

----- MENU TEXT -----

Before you can solve a problem you must define a new one or retrieve a previously defined one from disk. Please indicate the means you wish to use to obtain an active problem definition: (NOTE: To exit the program now, enter "0")

1. Load a previously defined problem from disk
2. Define a new problem

----- HELP TEXT -----

Before actual problem solving can be attempted, you must identify a problem to be solved together with its parameters. Currently, no problem has been defined or activated from disk. To exit the progra now, enter "0".

MENU NUMBER 6 STARTING RECORD NUMBER 30

----- MENU TEXT -----

Please select your next operation from the following menu:
(NOTE: To exit the progra now, enter "0")

1. Retrieve a different problem from disk
2. Define a new problem
3. Edit the currently active problem
4. Solve the currently active problem

----- HELP TEXT -----

To solve the problem currently in memory, select option 4. To change the currently active problem, select option 1, 2, or 3 as needed. To exit the program now, enter "0".

MENU NUMBER 7 STARTING RECORD NUMBER 36

----- MENU TEXT -----

Please enter the name of the problem that you now want to load from disk:

----- HELP TEXT -----

In order to retrieve a problem from disk, you must identify it with the name that was assigned to it when it was originally defined. Problem file names are from 2 to 8 characters in length and may contain no special characters or lower case letters. If you have forgotten your problem file name, you can obtain a list of all problems by leaving this program and typing "DIR B:". All problem file names will be of the form xxxxxxxx.PRB.

MENU NUMBER 8 STARTING RECORD NUMBER 42

-----MENU TEXT -----

A disk input/output error has occurred. Please make sure that the problem diskette is installed in drive2, the disk drive door is closed, and that you have entered the correct file name.

Would you like to try again?

1. Yes
2. No

----- HELP TEXT -----

If all of the above suggestions do not apply, the disk is bad and you should select option 2. In this event, you will have to redefine your problem. If one of the problems is applicable now, correct the problem and select option 1 to try again. For further information, refer to the users manual.

MENU NUMBER 9 STARTING RECORD NUMBER 48

----- MENU TEXT -----

DEFINE NEW MULTIOBJECTIVE PROBLEM

In this module, you will be defining a new multiobjective problem to be solved. You should now be prepared to enter the objective functions (up to four), and the constraints (up to ten). NOTE: If you make a mistake on some entry, you can easily correct and/or modify your problem later using the EDIT module. Also, your problem will automatically be saved to disk for later use.

----- HELP TEXT -----

HELP not applicable for this display...

MENU NUMBER 10 STARTING RECORD NUMBER 54

----- MENU TEXT -----

Please enter the name you wish to use for referring to this problem:
(NOTE: If you wish to stop now, enter "STOP")

----- HELP TEXT -----

The name you enter now will be used as a disk filename for saving your problem for later use. Filenames can be from two to eight upper-case characters, may contain no special characters, and must begin with a letter.

MENU NUMBER 11 STARTING RECORD NUMBER 58

----- MENU TEXT -----

The name you selected is already is use as a problem file name. Do you wish to redefine that problem now?

1. yes
2. no

----- HELP TEXT -----

If you enter 1, the existing file will be lost and you can proceed with your problem definition. If you enter 2, the existing file will be left unchanged, and you will be asked for a different file name.

MENU NUMBER 12 STARTING RECORD NUMBER   63

----- MENU TEXT -----

How many variables does your problem contain?

----- HELP TEXT -----

Problems may contain up to ten variables. These variables are the unknown quantities that are being solved for so as to optimize the objective function(s). If you make a mistake, you can add or delete variables later using the EDIT module.

MENU NUMBER 14 STARTING RECORD NUMBER   71

----- MENU TEXT -----

Please specify the objective function type:

1. Maximimize
2. Minimize

----- HELP TEXT -----

The purpose of the program is to optimize some objective subject to some constraints. You should now specify whether the the optimal condition for the current objective is a max or min.

MENU NUMBER 14 STARTING RECORD NUMBER   71

----- MENU TEXT -----

The menu contains variable data and is generated programmatically...

----- HELP TEXT -----

Enter the value for the objective coefficient for the specified variable. If the value is zero just hit the carriage return.

MENU NUMBER 15 STARTING RECORD NUMBER   74

----- MENU TEXT -----

Please enter the name or target quantity that this objective represents. If there are no more objectives for this problem, enter "/".

----- HELP TEXT -----

You may now enter a name to associate with this objective function. This name is optional and can be skipped by just hitting the carriage return.

MENU NUMBER 16 STARTING RECORD NUMBER 79

----- MENU TEXT -----

Please enter the name of the resource that this constraint represents. If there are no more constraints for the problem, enter "/".

----- HELP TEXT -----

You may now enter a name to associate with the constraint. The name is optional and may be avoided by hitting the carriage return with no input.

MENU NUMBER 17 STARTING RECORD NUMBER 84

----- MENU TEXT -----

The menu contains variable data and is generated programmatically...

----- HELP TEXT -----

Enter the value for the constraint coefficient for the specified variable. If the value is zero, just hit the return button.

MENU NUMBER 18 STARTING RECORD NUMBER 87

----- MENU TEXT -----

Please specify the type of relation applicable to this constraint:

1. less than or equal
2. equal
3. greater than or equal

----- HELP TEXT -----

Constraints specify a limiting quantity that applies to the allocation of the scarce resources. Please identify the type of relation that applies to the constraint.

MENU NUMBER 19 STARTING RECORD NUMBER 92

----- MENU TEXT -----

Please specify the right hand side value for this constraint:

----- HELP TEXT -----

You should now enter the limit applicable to the present constraint. If the value is zero, just hit the carriage return.

MENU NUMBER 20 STARTING RECORD NUMBER 95

----- MENU TEXT -----

EDIT PROBLEM

This function allows you to EDIT the problem that is currently loaded into program memory. Whenever an objective, constraint, or variable is to be referenced by number, the numbers are as they appear on the problem printout that can be obtained by entering "@P" (or "@" for screen) in response to any menu.

Please select from the following EDIT functions:
(NOTE: Enter "0" to exit the EDIT function)

----- HELP TEXT -----

HELP not applicable for this display....

MENU NUMBER 21 STARTING RECORD NUMBER 101

----- MENU TEXT -----

1. Add objective function
2. Delete objective function
3. Add constraint
4. Delete constraint
5. Add variable
6. Delete variable
7. EDIT objective coefficients
8. EDIT constraint coefficients
9. EDIT constraint right hand side

----- HELP TEXT -----

Select the EDIT function as needed. For some operations, you may have to delete a constraint or objective and the re-add it later. For example, to change a MAX objective to a MIN or to change an "=" constraint to a "<=" constraint.

MENU NUMBER 22 STARTING RECORD NUMBER 107

----- MENU TEXT -----

Please specify which of the above objectives you now wish to delete:
(NOTE: Enter "0" if you have changed your mind)

----- HELP TEXT -----

You have indicated that you want to delete one or more objective functions. You should now specify which one by entering its number. Remember, you may obtain a printout of your problem at any time by entering "@P", this printout lists the objective functions and their corresponding numbers.

MENU NUMBER 23 STARTING RECORD NUMBER 112

----- MENU TEXT -----

Please specify which of the above constraints you now wish to delete:
(NOTE: Enter "0" if you have changed your mind)

----- HELP TEXT -----

You have indicated that you want to delete one or more of the constraints from your currently active problem. You should now specify which one to delete by entering its number. Remember, you may obtain a printout of your problem at any time by entering "@P". This printout lists all of the constraints and their numbers.

MENU NUMBER 24 STARTING RECORD NUMBER 117

----- MENU TEXT -----

A variable has been added to your problem. The coefficient values for this variable for all objectives and constraints has been initialized to a value of 0.0. If you need to alter these values, you may do so using the EDIT functions to EDIT objectives and constraints accordingly.

----- HELP TEXT -----

HELP not applicable for this display...

MENU NUMBER 25 STARTING RECORD NUMBER 122

----- MENU TEXT -----

PRESS THE RETURN KEY WHEN READY TO CONTINUE...

----- HELP TEXT -----

To resume PCADA processing press the return key now.

MENU NUMBER 26 STARTING RECORD NUMBER 125

----- MENU TEXT -----

Please input the number of the variable you wish to delete:
(NOTE: Enter "0" if you have changed your mind)

----- HELP TEXT -----

You have indicated that you want to delete a variable from your problem. Please indicate now which variable you want to delete by entering its number. For example, to delete variable x3, enter 3. NOTE: The variables remaining in your problem will be uppacked. That is, if you delete x3, x4 will become x3 and x5 will become x4, etc.

MENU NUMBER 27 STARTING RECORD NUMBER 130

----- MENU TEXT -----

Please specify the number of the objective function whose coefficients you would like to EDIT: (NOTE: Enter "0" if you are done editing)

----- HELP TEXT -----

You have indicated that you would like to EDIT the coefficients of one or more of your objective functions. You should now specify which objective you want to EDIT. Remember, you can enter "@P" for a printout at any time.

MENU NUMBER 28 STARTING RECORD NUMBER 135

----- MENU TEXT -----

Please specify the number of the coefficient that you would like to change:
(NOTE: When done editing, enter "0")

----- HELP TEXT -----

You are in the process of editing constraint or objective coefficients. You should now indicate which coefficient you want to change. For a review of your problem, enter "@P" to recieve a problem printout. To stop editing this objective (constraint) enter "0".

MENU NUMBER 29 STARTING RECORD NUMBER 140

----- MENU TEXT -----

Please specify the new coefficient value:

----- HELP TEXT -----

You are in the process of editing a constraint or objective. You should now enter the new value for the specified coefficient.

MENU NUMBER 30 STARTING RECORD NUMBER 143

----- MENU TEXT -----

Please specify the number of the constraint whose coefficients you would like to EDIT: (NOTE: If you are done editing coefficients, enter "0")

----- HELP TEXT -----

You are in the process of editing constraint coefficients. You should now indicate which constraint you wish to EDIT by entering its number. To refresh your memory about constraint numbers, you can now request a printout of your problem by entering "@P".

MENU NUMBER 31 STARTING RECORD NUMBER 148

----- MENU TEXT -----

Please specify the number of the constraint whose right hand side you would like to EDIT: (NOTE: If you are done editing right hand sides, enter "0")

----- HELP TEXT -----

You have indicated that you want to alter one or more constraint right hand side values. You should now identify which constraint you want to EDIT by entering its number. Recall, that you can obtain a printout of your problem showing constraints and their numbers by entering "@P".

MENU NUMBER 32 STARTING RECORD NUMBER 154

----- MENU TEXT -----

Please enter the new right hand side value:

----- HELP TEXT -----

You have indicated that you want to change constraint right hand sides and should now enter the new right hand side value for the constraint you have indicated. NOTE: Right hand side values MUST be positive. If yours is negative, you must convert the constraint by multiplying it by -1. This operation will change a less than relation to a greater than relation as well as changing the sign on all of the coefficients.

MENU NUMBER 33 STARTING RECORD NUMBER 160

----- MENU TEXT -----

Please specify the right hand side value for this constraint:

----- HELP TEXT -----

You should now enter the limit applicable to the present constraint. If the value is zero, just hit the return button. NOTE: Negative right hand side values are not allowed. If your constraint has a negative right hand side, you must convert it by multiplying it by -1. In addition to changing all coefficients, this operation will change the relation from less than to greater than and vice-versa.

MENU NUMBER 34 STARTING RECORD NUMBER 166

----- MENU TEXT -----

Please specify which problem solving technique you now wish to use to solve the current problem (enter "0" to exit):

1. The weighting technique
2. The constraint technique

----- HELP TEXT -----

You should specify which of the two problem solving techniques you wish to use. If you are unsure which one is appropriate, refer to the users manual or to the instructional module of this program. In short, the weighting technique is faster, and the constraint technique requires some advance knowledge about the objective function ranges.

MENU NUMBER 35 STARTING RECORD NUMBER 172

----- MENU TEXT -----

THE WEIGHTING TECHNIQUE

The weighting technique solves the problem by assigning weights to each of the separate objectives, forming a new objective, and optimizing it. As a result, the program will generate a "set" of solutions with each one corresponding to a different weight combination. These "non-dominated" solutions will be output and can then be compared to determine which one(s) is(are) appropriate.

----- HELP TEXT -----

HELP not applicable for this display...

MENU NUMBER 36 STARTING RECORD NUMBER 178

----- MENU TEXT -----

Please specify the weight increment to be used for this solution:
(NOTE: Enter "0" to exit the weighting technique)

1. 1.000
2. 0.500
3. 0.333
4. 0.250
5. 0.200
6. 0.100
7. 0.050

----- HELP TEXT -----

The smaller the increment you select, the longer the program will run, and the more complete the solution set will be.

MENU NUMBER 37 STARTING RECORD NUMBER 183

----- MENU TEXT -----

Please select from the following weighting technique solution options:
(NOTE: Enter "0" to exit the weighting technique)

1. Solve with specific objective weights
2. Solve with a range of weights on all objective functions

----- HELP TEXT ------

Option 1 will optimize the combined objective only for the 1 set of weights that you will supply. Option 2 will iteratively solve the problem for a range of weights on each objective function. Initially, you should use option 2, use option 1 when you want to investigate some particular weight combination(s).

MENU NUMBER 38 STARTING RECORD NUMBER 189

----- MENU TEXT -----

Menu is generated programmatically...

----- HELP TEXT -----

You have specified that you wish to solve the problem using some particular weight combination. You should now specify the weights you want. Note: weights must be from 0 to 1, but they need NOT neccessarily sum to 1.0. If you enter .5, .5, and .5 for each of three objectives, the program will accept it and weight each of the three objectives equally (but not 50% each).

MENU NUMBER 39 STARTING RECORD NUMBER 194

----- MENU TEXT -----

THE CONSTRAINT TECHNIQUE

The constraint technique solves multiobjective problems by making all but one of the objectives into equality constraints and optimizing the one remaining objective. The right hand sides of the "new" constraints are bounded by the user as to the acceptable upper and lower limits. These right hand sides are then varied between the limits to obtain a set of acceptable solutions.

----- HELP TEXT -----

HELP not applicable for this display...

MENU NUMBER 40 STARTING RECORD NUMBER 200

----- MENU TEXT -----

Please enter the upper acceptable limit for the above objective:

----- HELP TEXT -----

You should now specify the highest value for the specified objective that you are willing to accept or that you can reasonably expect. You should try to insure that the value you input is in range, otherwise, computing time will be wasted and the final solution will be less complete.

MENU NUMBER 41 STARTING RECORD NUMBER 205

----- MENU TEXT -----

Please enter the lowest acceptable limit for the objective function specified above:

----- HELP TEXT -----

You should now input the lowest value that is acceptable and/or attainable for the specified objective function. You should try hard to specify "close" values otherwise, computing time will be wasted and the final solution may not be as complete as otherwise possible.

MENU NUMBER 42 STARTING RECORD NUMBER 210

----- MENU TEXT -----

Please specify the number of steps for solving the problem in going from the constraint lower limit to the upper limit:

----- HELP TEXT -----

The higher the number of steps, the more time the solution will require but the more non-dominated solutions will be found. A value of 10 is a good first try.

MENU NUMBER 43 STARTING RECORD NUMBER 214

----- MENU TEXT -----

Please enter the number of the solution that you would like to see:

----- HELP TEXT -----

Solution numbers correspond to the objective summary display. The solution you select will be displayed in detail, either to the display screen or the printer as you specify. Also, following the display, you will still have an opportunity to examine other solutions. However, once you exit the solution display module, solutions are lost.

MENU NUMBER 44 STARTING RECORD NUMBER 219

----- MENU TEXT -----

----> Your problem had no feasible solutions <----

Check for error in formulation and/or erroneous constraints/objective functions

WHEN READY TO CONTINUE, PRESS THE RETURN KEY

----- HELP TEXT -----

Your problem solution is complete and no feasible (or unbounded) solutions were found. Check problem accuracy and try again and/or adjust weights/limits.

MENU NUMBER 45 STARTING RECORD NUMBER 224

----- MENU TEXT -----

At least one unbounded solution was obtained. This suggests a potential problem formulation error. Please check your problem carefully. The unbounded solution first occurred under the following conditions:

----- HELP TEXT -----

HELP not applicable for this display...

MENU NUMBER 46 STARTING RECORD NUMBER 228

----- MENU TEXT -----

----> Your single objective problem is unbounded. <----

Check for error in formulation and/or erroneous constraints/objective functions.

WHEN READY TO CONTINUE, PRESS THE RETURN KEY

----- HELP TEXT -----

Your problem (which consists of a single objective function) is unbounded. Since unbounded solutions rarely occur in real life, your problem is probably incorrectly formulated. Please check your problem carefully and try again.

MENU NUMBER 47 STARTING RECORD NUMBER 233

----- MENU TEXT -----

Please specify your choice for examining problem solutions:
(NOTE: Enter "0" to exit the solution display module)

1. Objective value summary -- all solutions
2. Detailed information -- one specific solution
3. Detailed information -- all solutions

----- HELP TEXT -----

Select which display option you desire. You will be given the choice to route answers to either the printer or screen. If you specify "0", you will have to resolve the problem in order to see the solutions again.

MENU NUMBER 48 STARTING RECORD NUMBER 239

----- MENU TEXT -----

Would you like your solution outputs to include the objective weights that were used to obtain the corresponding solution?

----- HELP TEXT -----

If you would like to see the objective weights that were used to obtain the various solutions select option 1. NOTE: It is probable that each solution is obtainable by using many different weight combinations.

MENU NUMBER 49 STARTING RECORD NUMBER 244

----- MENU TEXT -----

Please indicate the desired output device for the solutions:

1. Display screen
2. Printer

----- HELP TEXT -----

Solutions can be presented either for quick reference on the display screen, or for permanent hardcopy on the printer. If you are unsure, select 1 now to review the answers and then if desired select 2 later.

MENU NUMBER 50 STARTING RECORD NUMBER 248

----- MENU TEXT -----

>> TO PRINT THE SOLUTIONS <<

1. Turn on the printer
2. Set the printer "on-line"
3. Align the paper

*** WHEN READY TO CONTINUE, PRESS THE CARRIAGE RETURN ***

----- HELP TEXT -----

You have requested a printout of problem solutions. Follow menu instructions and then press the return key.

MENU NUMBER 51 STARTING RECORD NUMBER 252

----- MENU TEXT -----

----> NOTE: There were more than the program limit of 30 solutions <----

The first 30 NON-DOMINATED solutions are presented in what follows. To get the rest, you will have to adjust your problem and solve it again. Refer to the users manual for further information.

----- HELP TEXT -----

HELP not applicable for this display....

VITA

Greg R. White was born in Pontiac, Michigan on 15 April 1951 and graduated from high school in Royal Oak, Michigan in 1969. He attended Michigan State University and graduated in 1973 with a Bachelor of Science degree in Computer Science.

Following graduation, Mr. White was employed by The Boeing Company in Seattle, Washington as a computer specialist. His assignments have involved all phases of software development including design, implementation and testing for both large and small scale computer systems. In 1981, Mr. White received the degree of Master of Business Administration from Seattle University. In June of 1983 he entered the Air Force Institue of Technology.

Mr. White is married to the former Cheryl Streetman of Berkley, Michigan and has one daughter, Sara age 4.

Permanent address: 12052 SE 212th Ct
Kent, Washington
98031

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GSO/OS/84D-8	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) Greg R. White, B.S.	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 84 Dec 14 15. PAGE COUNT 165
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
12	02	
09	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Multiobjective Linear Programming, Microcomputers, Linear Programming, Optimization, Personal Computers.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: Personal Computer Aided Decision Analysis

Thesis Advisor: Mark W. Mekaru, Lt. Col., Ph.D., USAF
Reader: James K. Feldman, Major, Ph.D., USAF

Approved for public release: IAW AFR 190-1.
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AIC)
Wright-Patterson AFB OH 45433

The increasing complexity of today's business and military decisions demand informed decision making at all levels of management. Such decision making must be fully supported by timely and accurate analysis. Computers are well-suited for such analysis. Unfortunately, the large mainframe computers are not flexible or responsive enough for use by most managers in a timely manner.

The growing popularity, presence, and capability of microcomputers represents a new opportunity for operations research. These small, low-cost machines can provide

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code) 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

(Block 19 continued)
much of the computer support needed for decision making by managers and analysts provided that the necessary software tools are developed.

This study was undertaken to provide a user-oriented decision analysis tool which exploits the advantages of personal computers. Of the many useful quantitative techniques available, the weighting and constraint techniques of multiobjective decision analysis were selected and implemented.

END

FILMED

5-85

DTIC